6683501 2582260

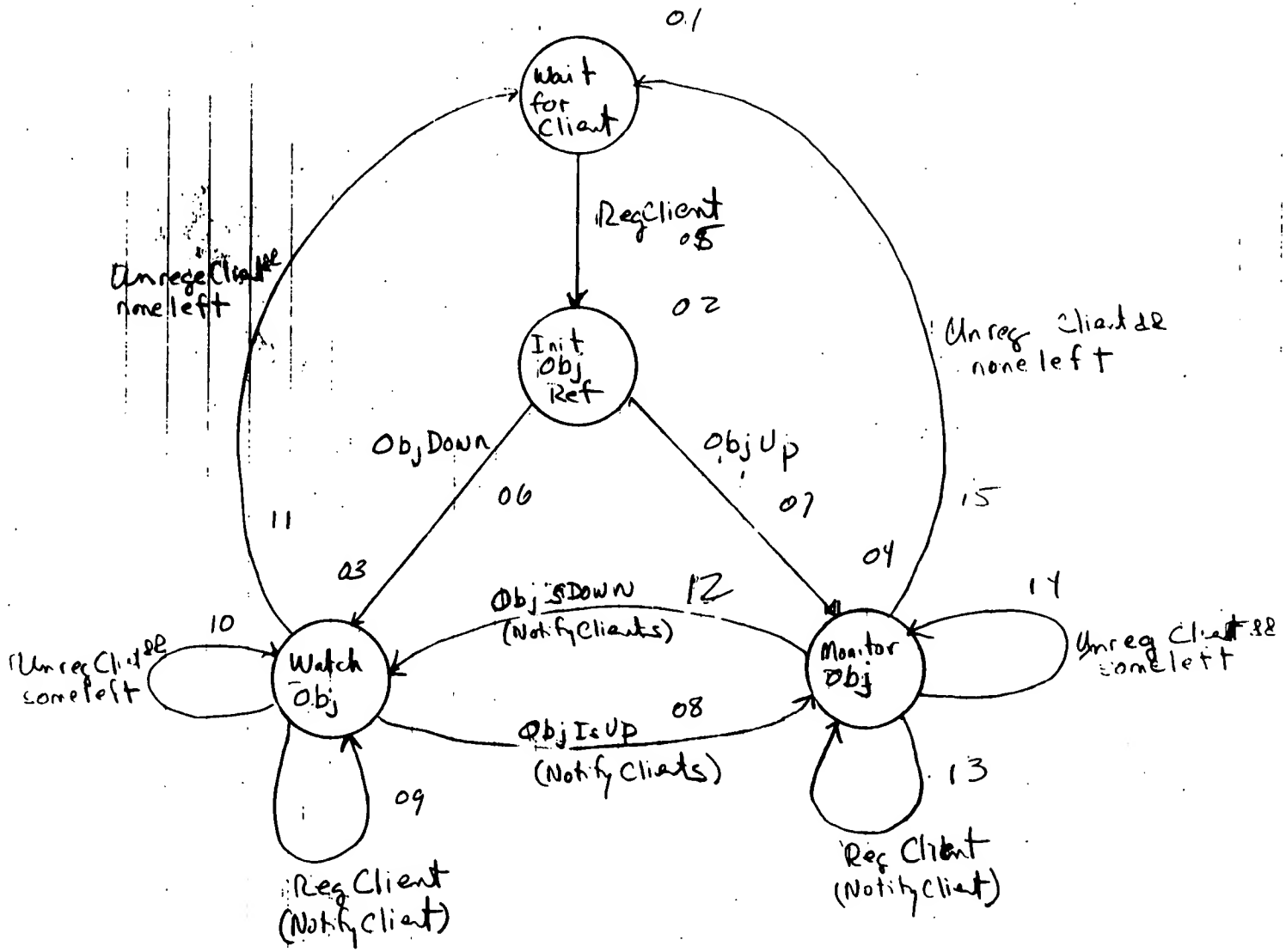



Fig 1

6682501 2582260

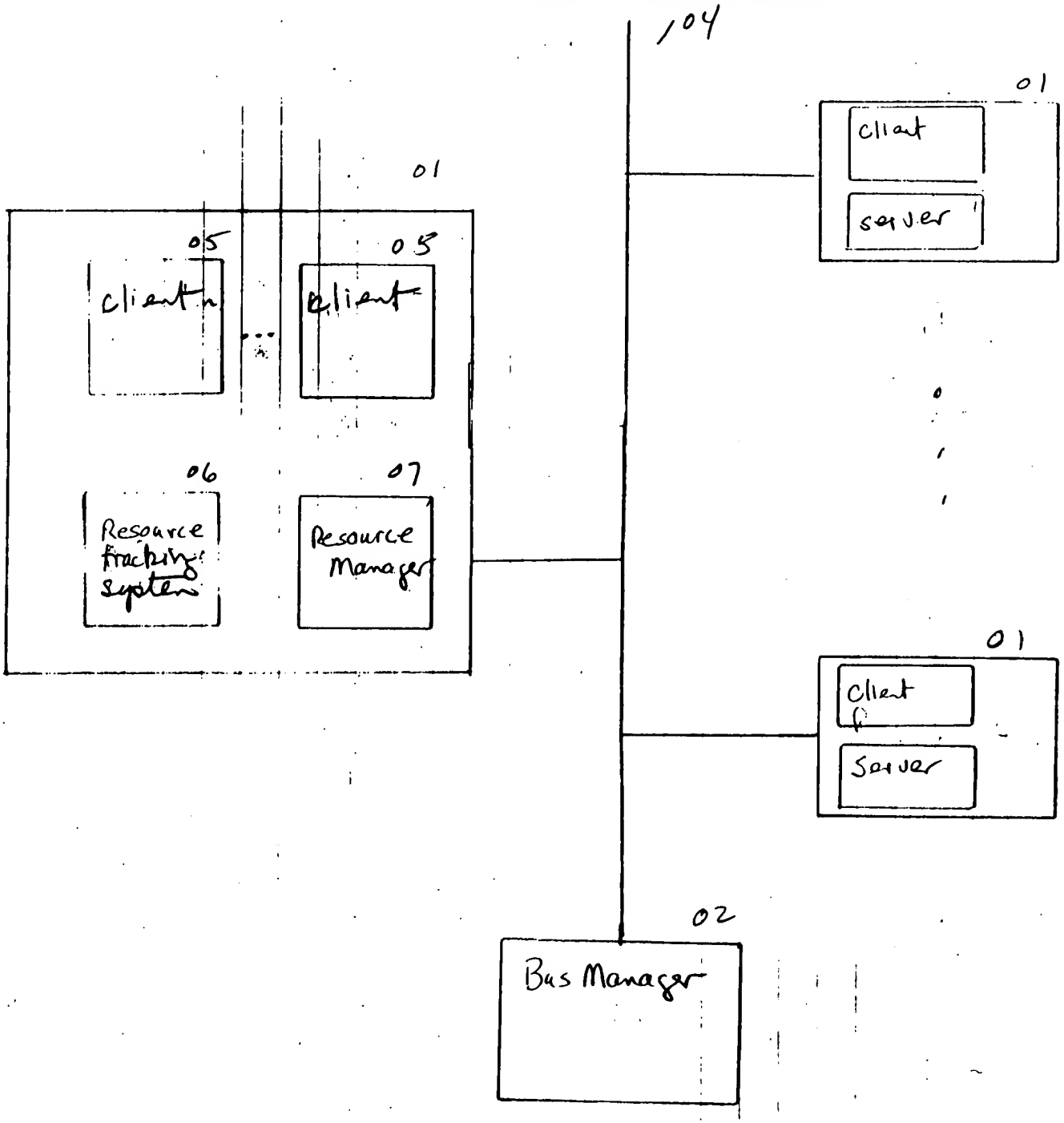
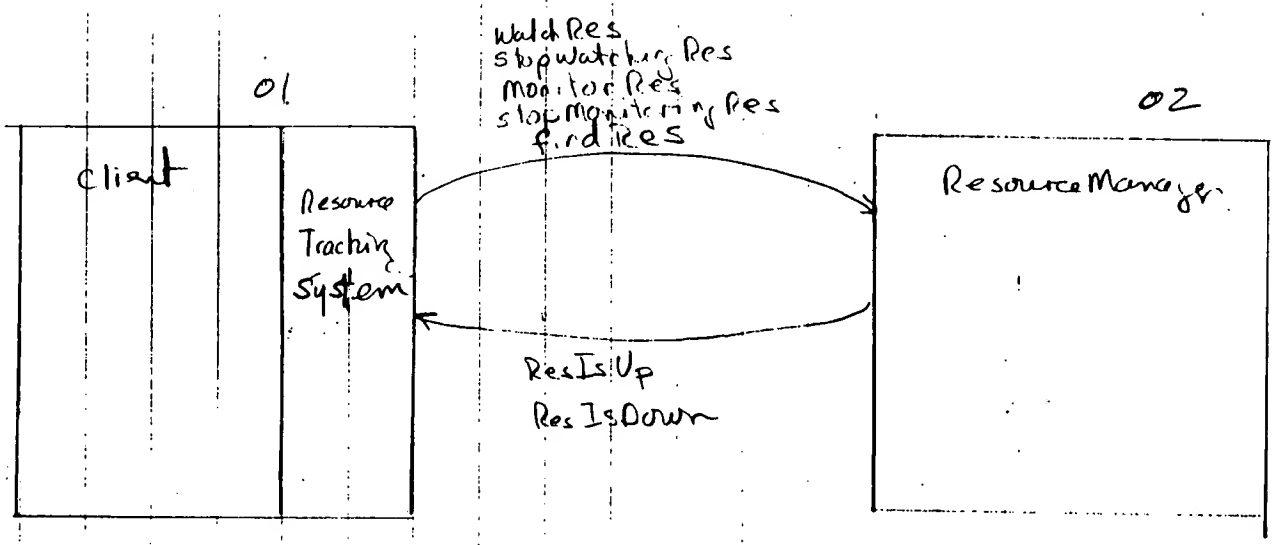



Fig 2


01
Client
Resource
Tracking
System
watchRes
stopWatching Res
monitorRes
stopMonitoring Res
findRes
02
Resource Manager
ResIsUp
ResIsDown

2A

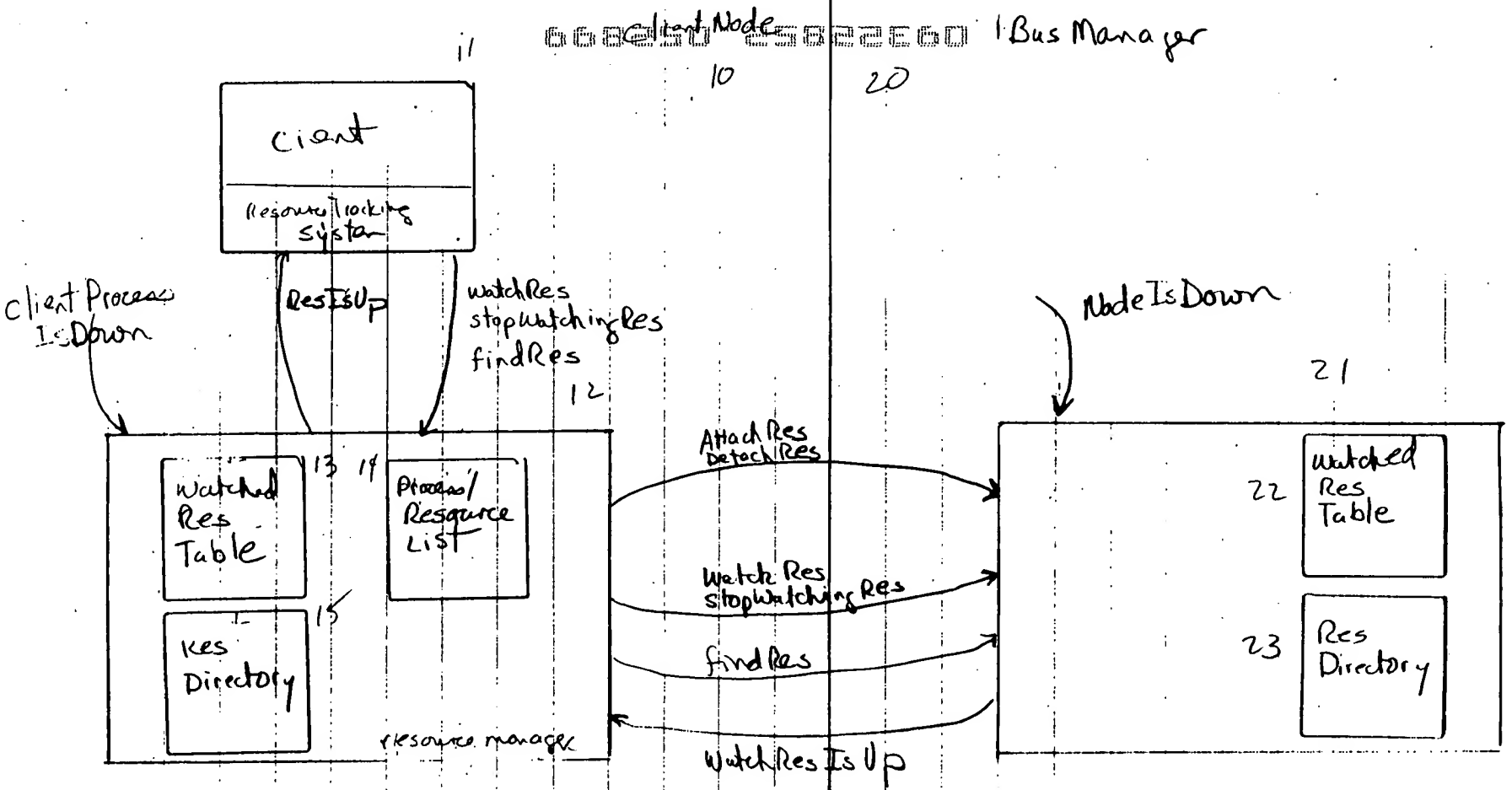

Client Node
Bus Manager
11
10
20
Client
Resource locking system
Client Process Is Down
Res Is Up
watchRes
stopWatchingRes
findRes
12
Node Is Down
21
13
14
Watched Res Table
Process/ Resource List
15
Res Directory
resource manager
Attach Res
Detach Res
22
Watched Res Table
Watch Res
stopWatching Res
23
Res Directory
findRes
Watch Res Is Up

Fig 2B

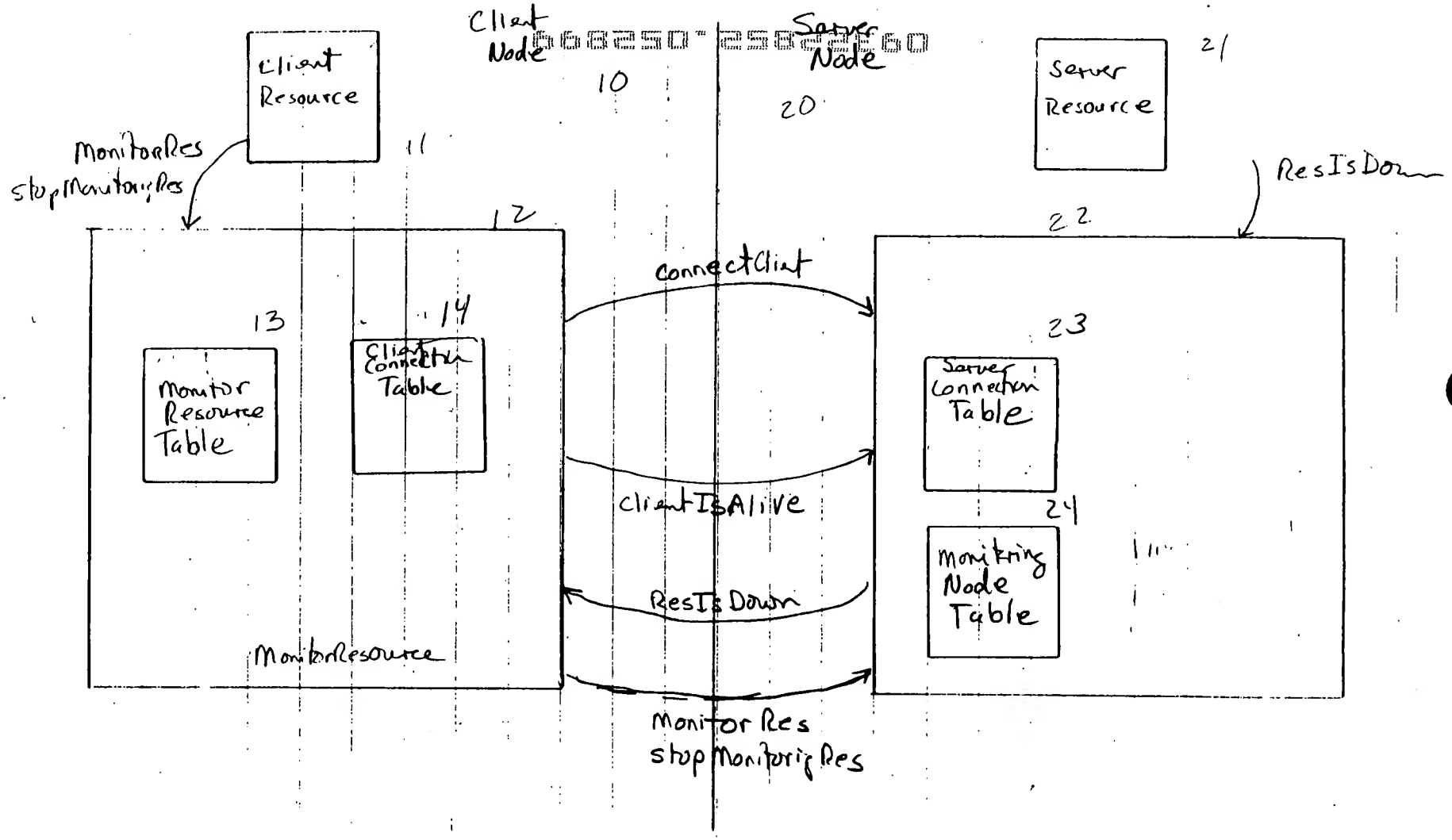
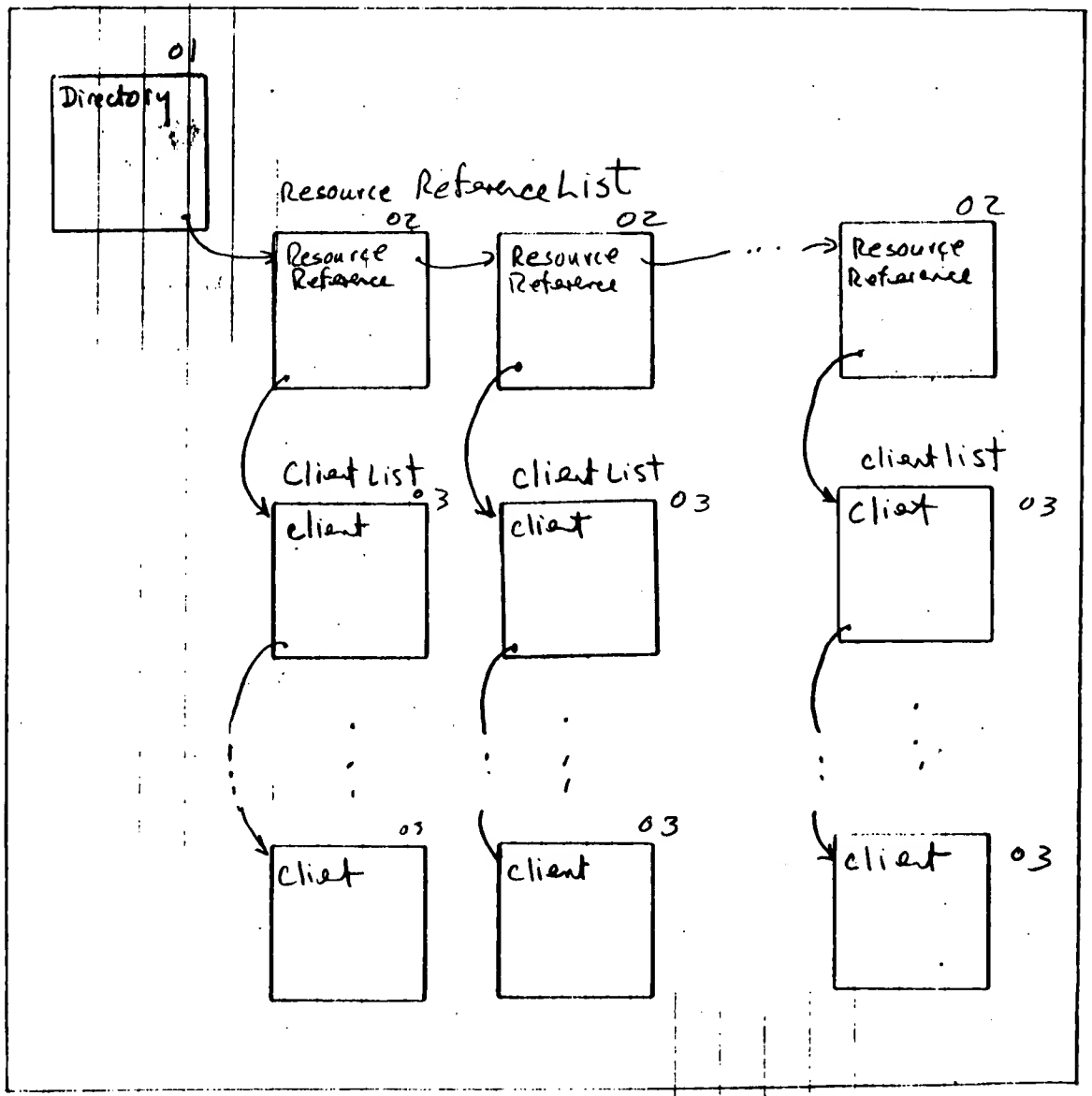

Client Node
Server Node
10
20
Client Resource
11
MonitorRes
stopMonitoringRes
12
13
14
Monitor Resource Table
Client Connection Table
MonitorResource
Server Resource
21
ResIsDown
22
23
Server Connection Table
24
Monitring Node Table
ConnectClient
clientIsAlive
ResIsDown
MonitorRes
stopMonitoringRes

Fig. 2C


01
Directory
Resource Reference List
02
Resource Reference
02
Resource Reference
02
Resource Reference
Client List
03
client
Client List
03
client
client list
03
Client
03
Client
03
client
03
client
03
client

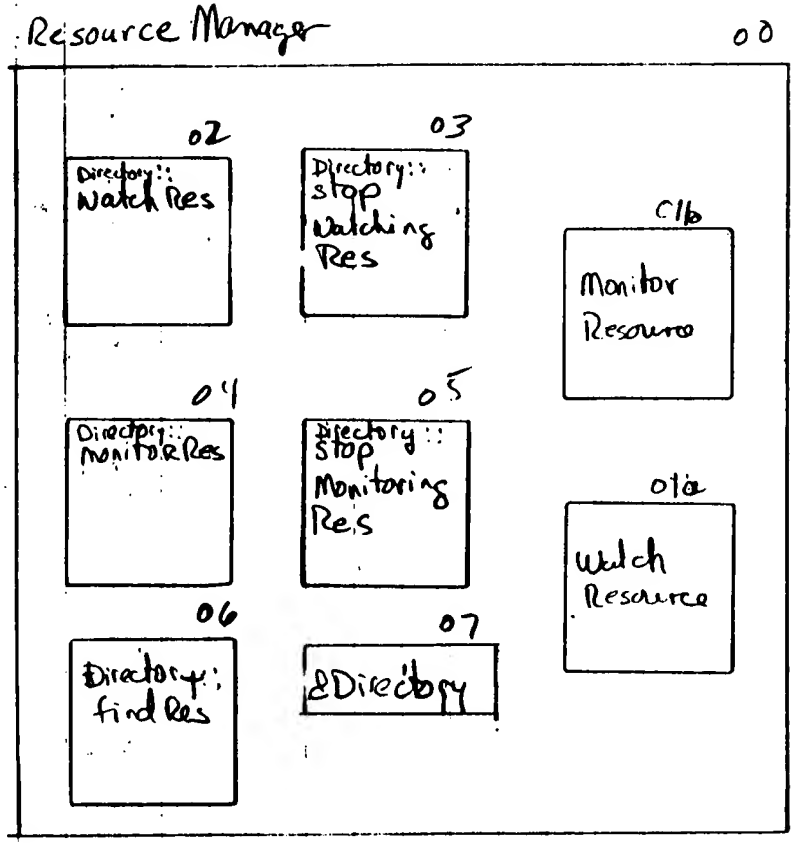

Resource Manager
00
02
Directory:: Watch Res
03
Directory:: Stop Watching Res
01b
Monitor Resource
04
Directory:: Monitor Res
05
Directory:: Stop Monitoring Res
01a
Watch Resource
06
Directory:: find Res
07
Directory

Fig 4

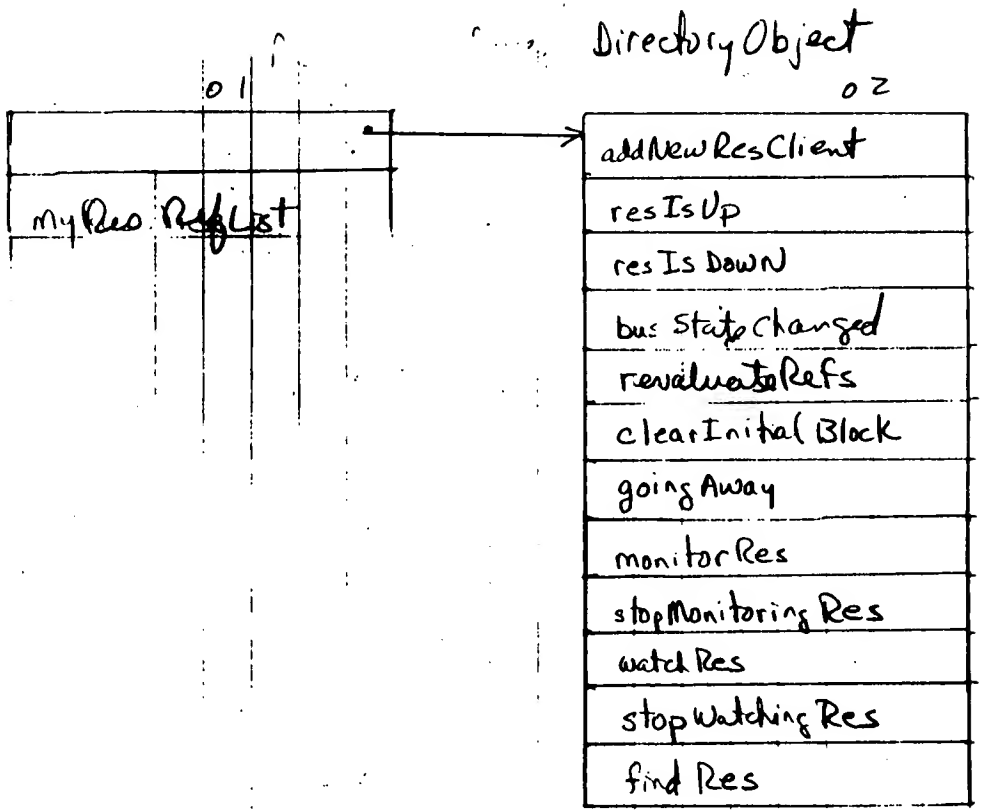

01
myResRefList
DirectoryObject
02
addNewResClient
resIsUp
resIsDown
busStateChanged
revaluateRefs
clearInitalBlock
goingAway
monitorRes
stopMonitoringRes
watchRes
stopWatchingRes
findRes

Fig 5

6688250 2582260

Resource Reference Object

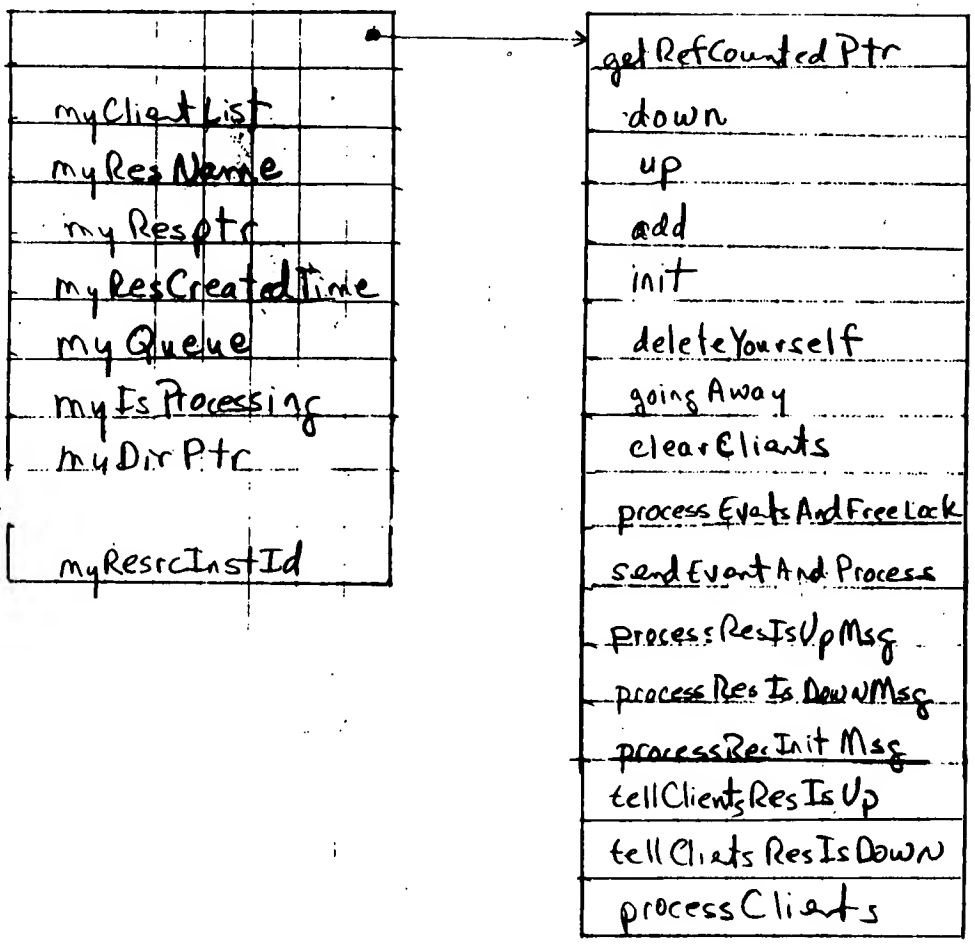



Fig 6

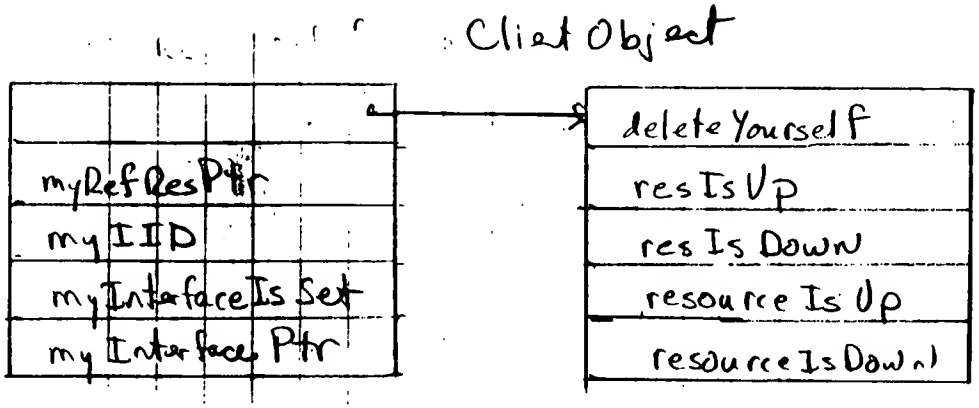

Client Object
myRefResPtr
myIID
myInterfaceIsSet
myInterfacePtr
deleteYourself
resIsUp
resIsDown
resourceIsUp
resourceIsDown

Fig 7

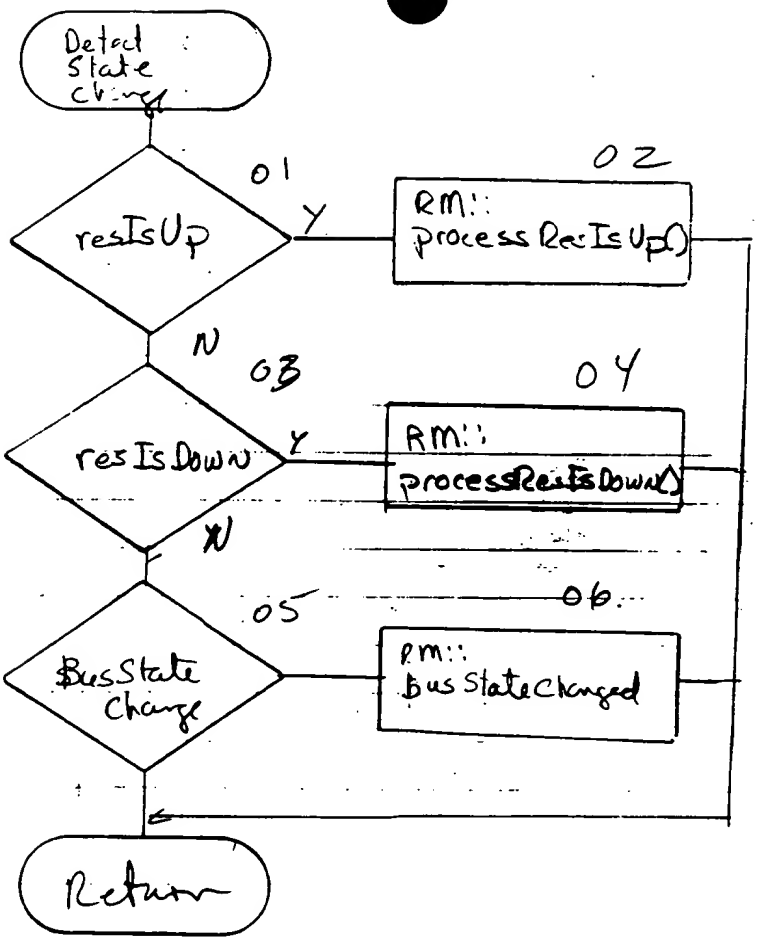

Detect State Change
01
resIsUp
Y
02
RM::
processResIsUp()
N
03
resIsDown
Y
04
RM::
processResIsDown()
N
05
BusState Change
06
RM::
BusStateChanged
Return

Fig 8

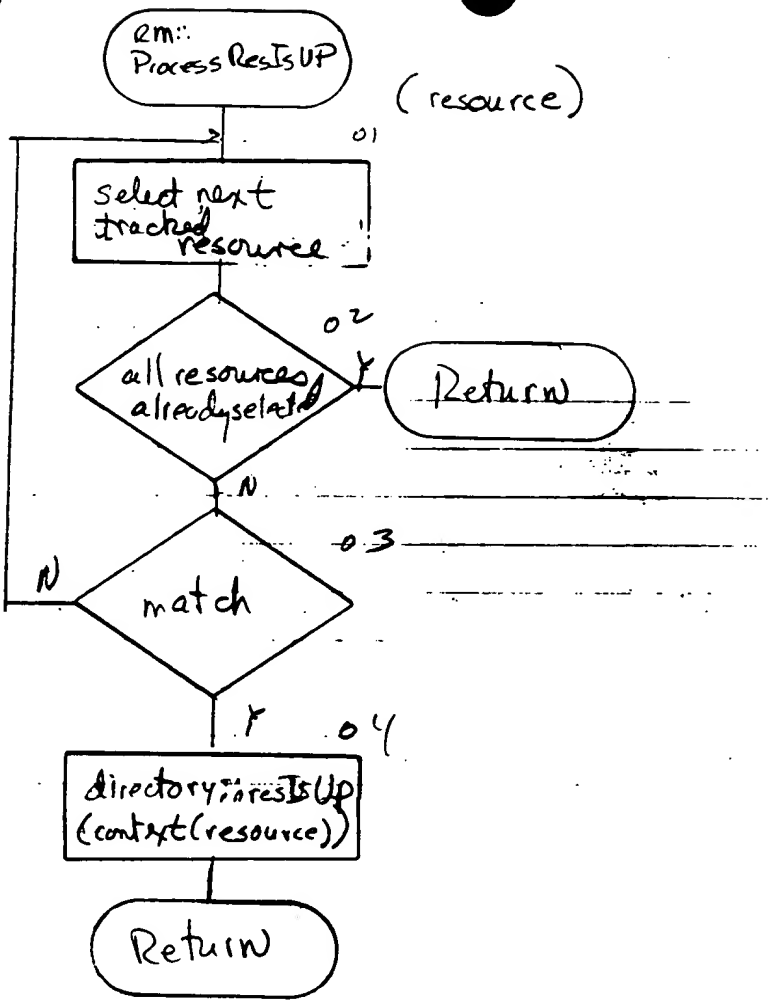

Rm::
ProcessResIsUP
(resource)
01
select next
tracked
resource
02
all resources
already selected
Y
Return
N
03
N
match
Y
04
directory::resIsUp
(context(resource))
Return

Fig 9

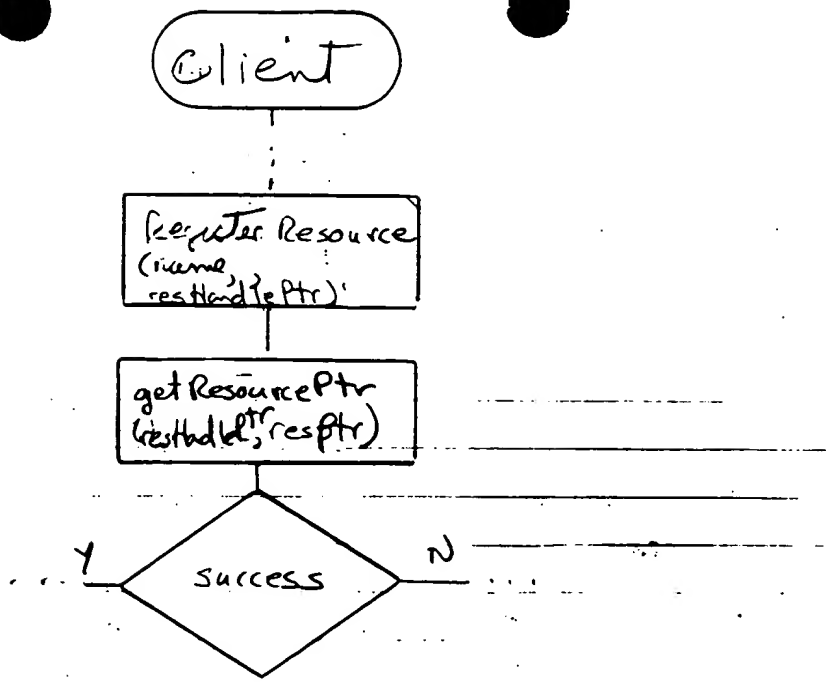

Client
Register Resource
(name,
resHandlePtr)
getResourcePtr
(resHandle, resPtr)
Y
success
N

Fig 10

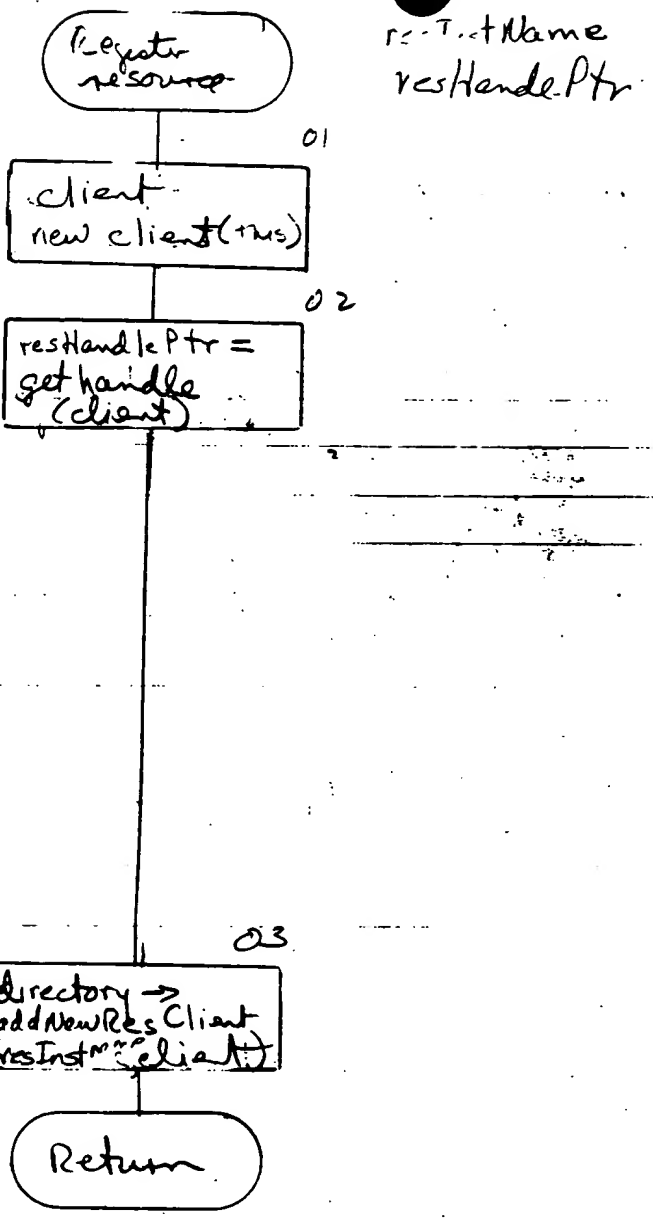

Register resource
resHandlePtr
01
client = new client(this)
02
resHandlePtr = getHandle (client)
03
Directory → addNewRes Client (resInstPtr, client)
Return

Figure 11

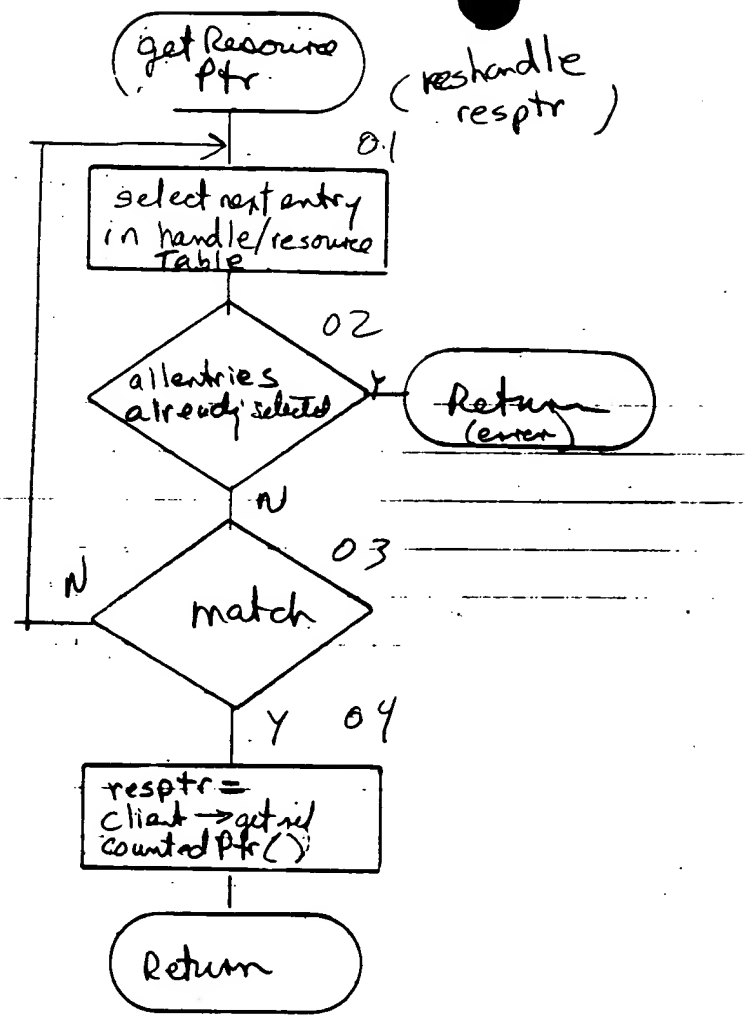

getResource Ptr
(reshandle resptr)
01
select next entry in handle/resource Table
02
all entries already selected
Y
Return (error)
N
03
N
match
Y
04
resptr = client → getrid countedPtr()
Return

Fig 12

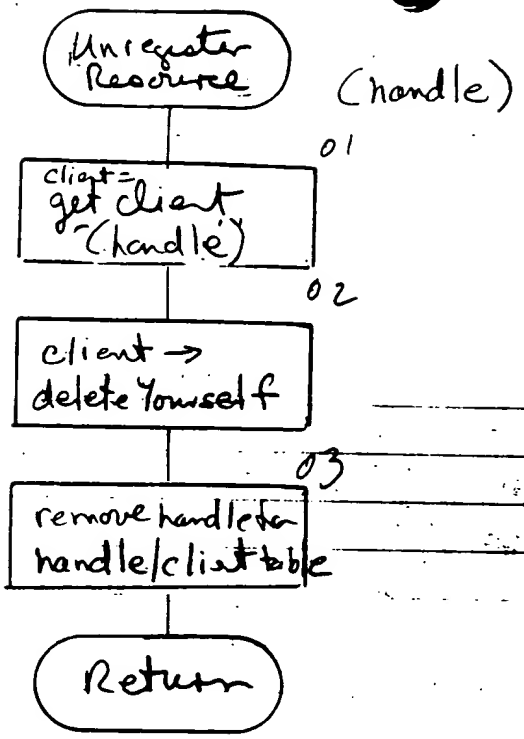

Unregister Resource
(handle)
01
client = get client (handle)
02
client → delete Yourself
03
remove handle from handle/client table
Return

Fig 13

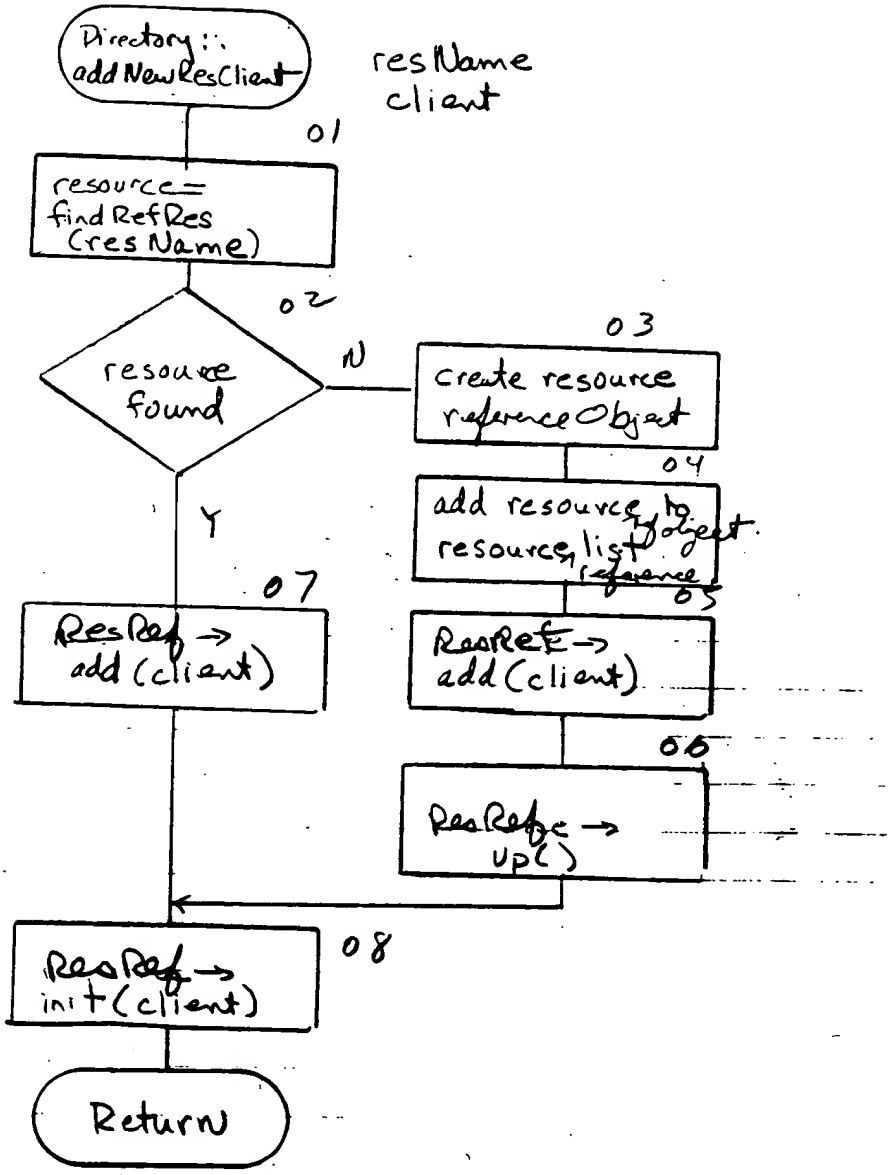

Directory::
addNewResClient
resName
client
01
resource=
findRefRes
(resName)
02
resource
found
N
03
create resource
referenceObject
04
add resource to
resource list
05
Y
07
ResRef →
add(client)
ResRef →
add(client)
06
ResRef →
up()
ResRef →
init(client)
08
Return

Fig 14

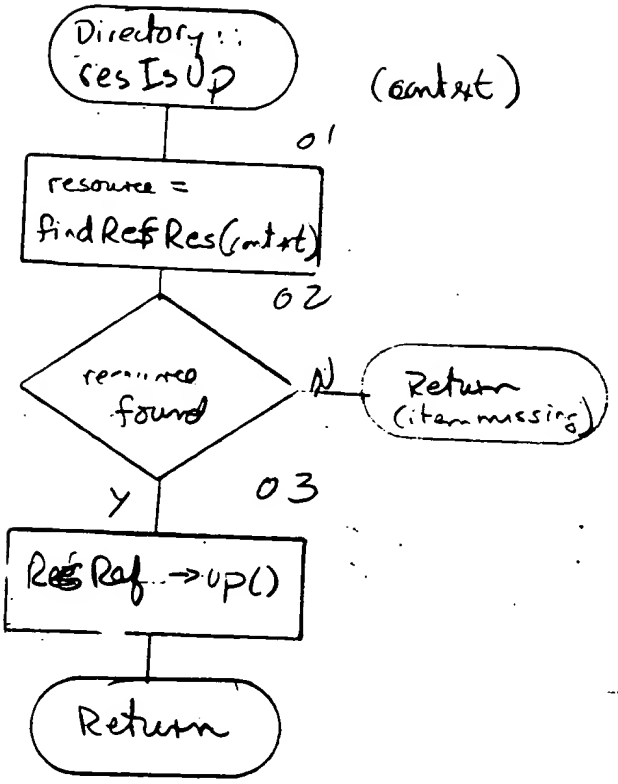

Directory::
resIsUp
(context)
01
resource =
findRefRes(context)
02
required
found
N
Return
(item missing)
Y
03
Ref->up()
Return

Fig 15

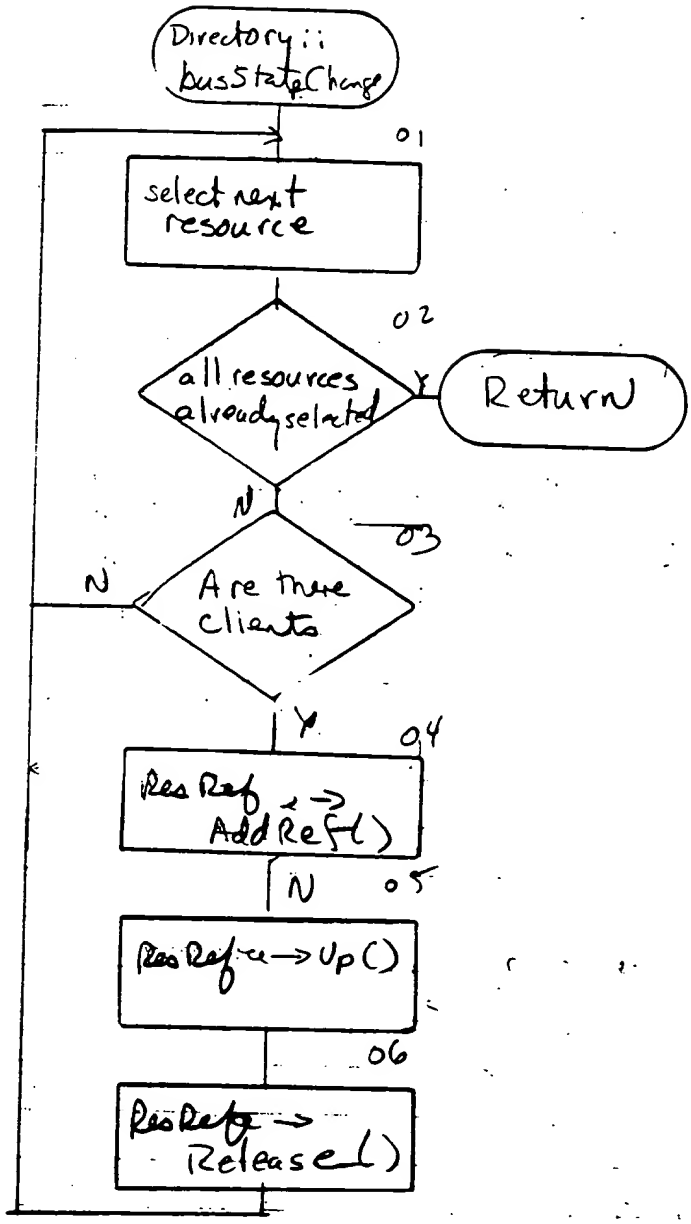

Directory::
busStateChange
01
select next
resource
02
all resources
already selected
Y
Return
N
03
N
Are there
clients
Y
04
ResRef->
AddRef()
N
05
ResRef->Up()
06
ResRef->
Release()

Fig 16

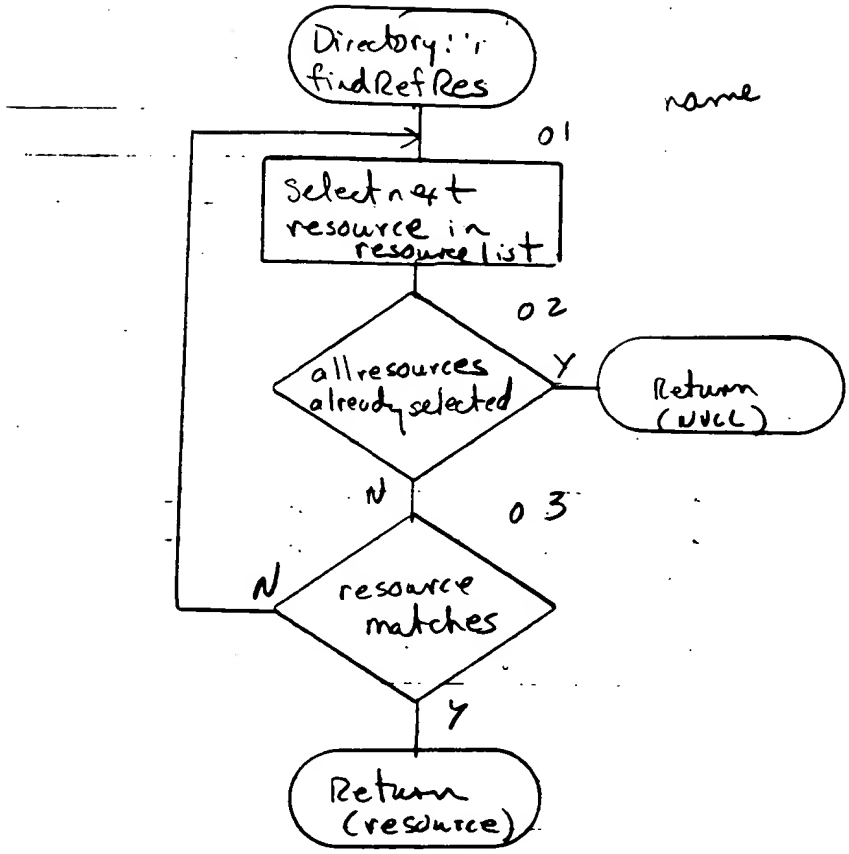

Directory::
findRefRes
name
01
Select next resource in resource list
02
all resources already selected
Y
Return (NULL)
N
03
N
resource matches
Y
Return (resource)

Fig 17

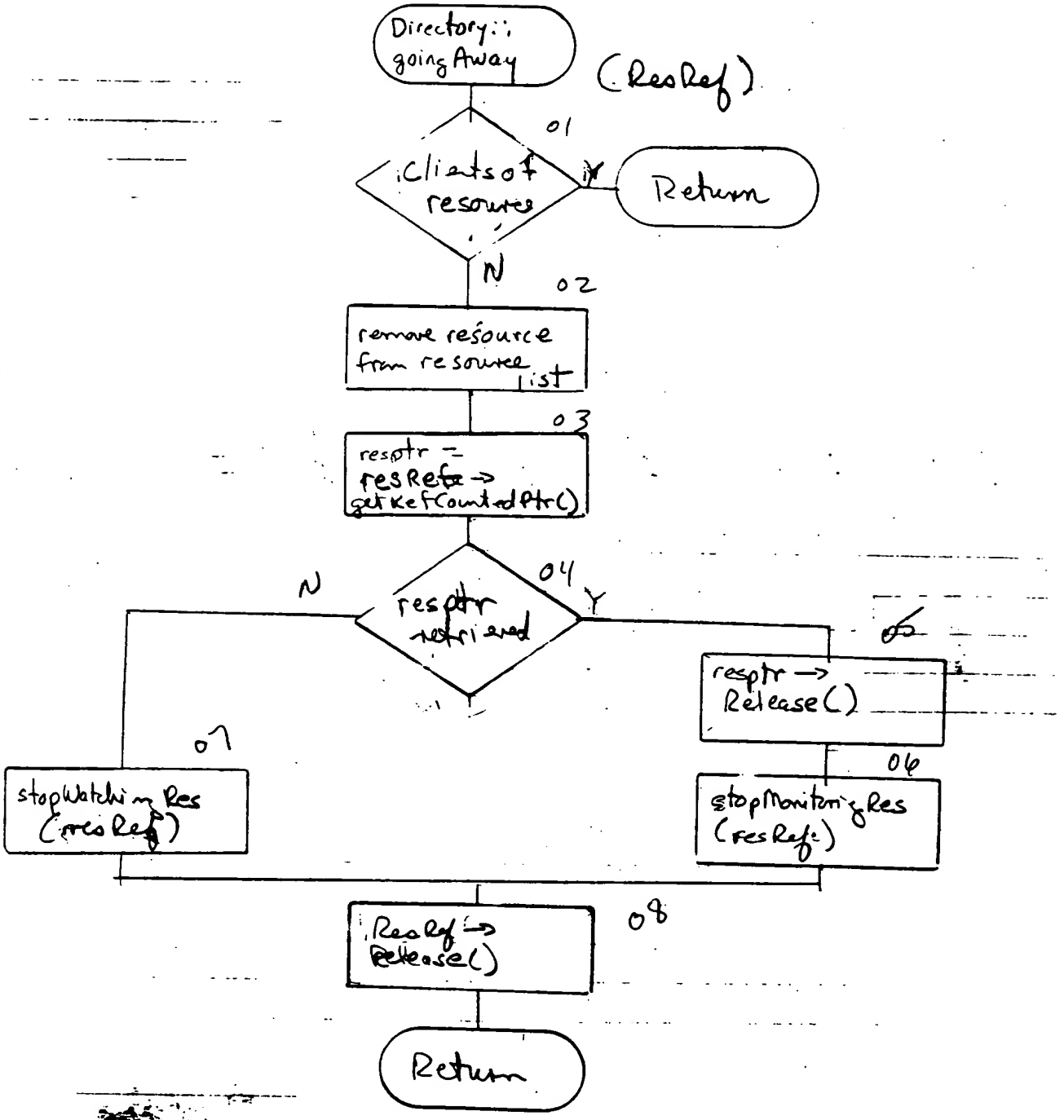

Directory::
goingAway
(ResRef)
01
:Clients of
resource
Y
Return
N
02
remove resource
from resource list
03
resptr =
resRef->
getRefCountedPtr()
04
N
resptr
retrieved
Y
05
resptr->
Release()
06
stopMonitoring Res
(resRef)
07
stopWatching Res
(resRef)
08
:ResRef->
Release()
Return

Figure 18

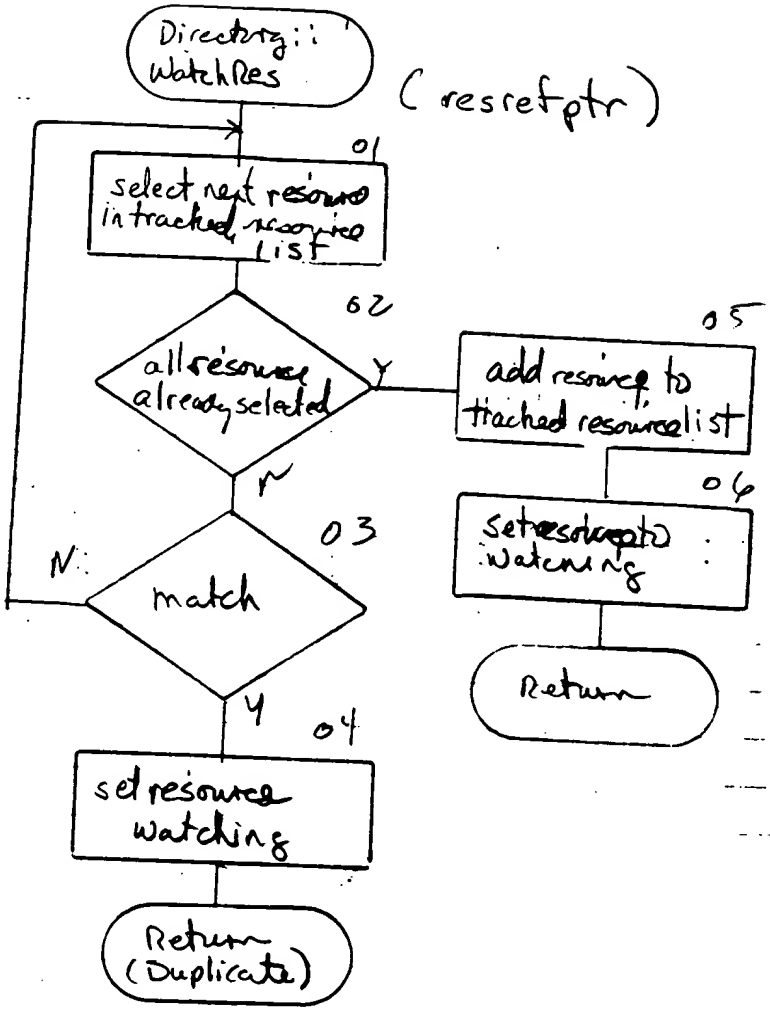

Directory::
WatchRes
(resrefptr)
01
select next resource
in tracked resource
list
02
all resources
already selected
Y
05
add resource to
tracked resource list
06
Set resource to
watching
Return
N
03
match
Y
04
set resource
watching
Return
(Duplicate)

Fig 19

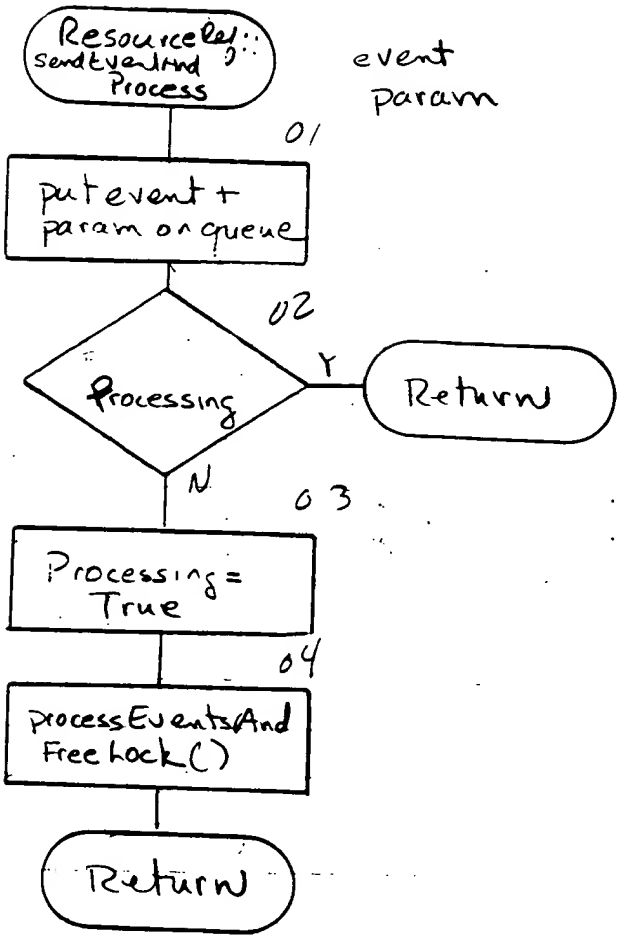

ResourceRel::
sendEventAnd
Process
event
param
01
put event +
param on queue
02
Processing
Y
Return
N
03
Processing =
True
04
processEventsAnd
FreeLock()
Return

Fig. 20

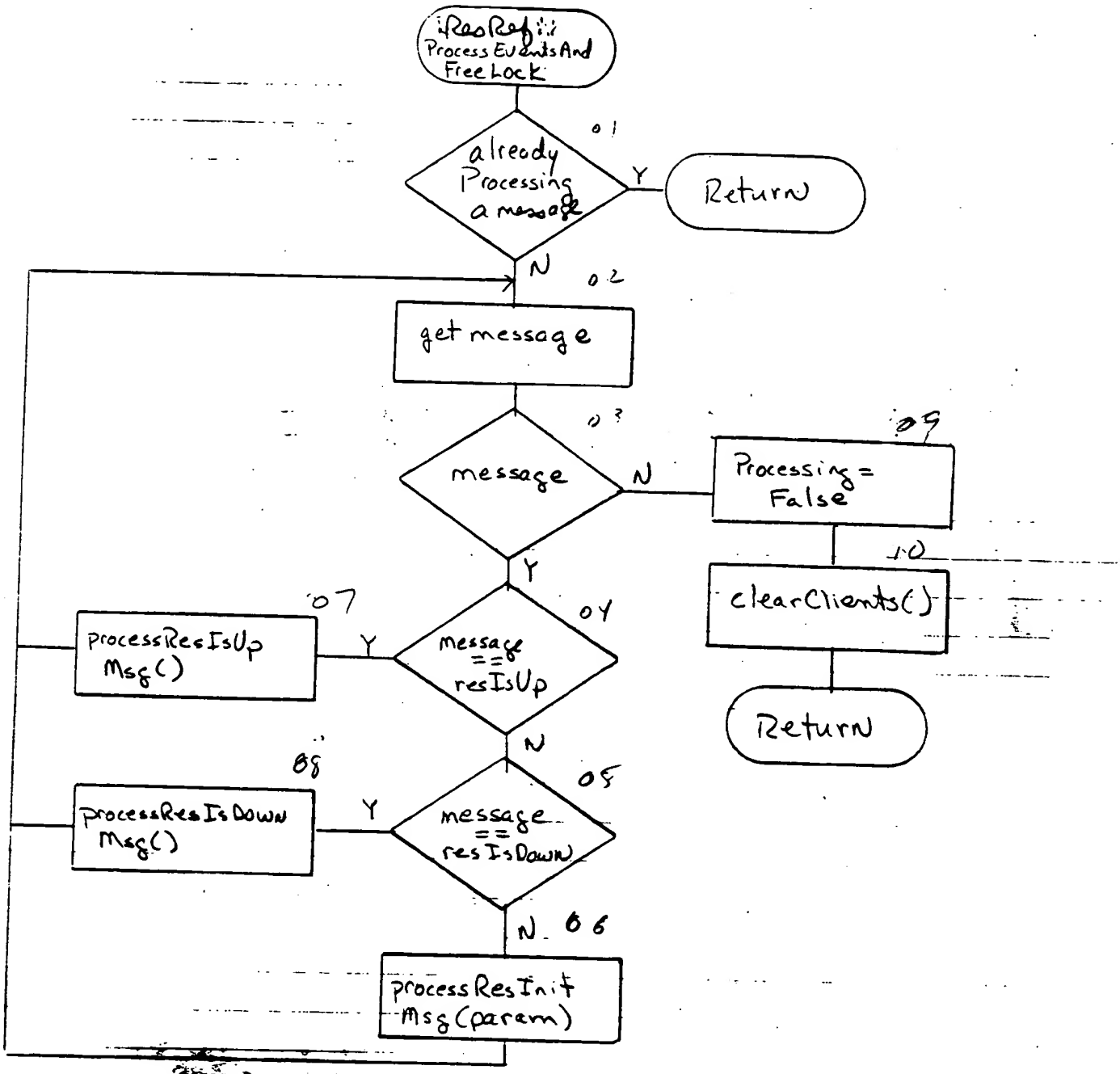

ResRef::
Process Events And
Free Lock
already
Processing
a message
Y
Return
01
N
02
get message
message
03
N
Processing =
False
09
10
clearClients()
Return
Y
04
message
==
resIsUp
Y
07
processResIsUp
Msg()
N
05
message
==
resIsDown
Y
08
processResIsDown
Msg()
N
06
processResInit
Msg(param)

Fig 2

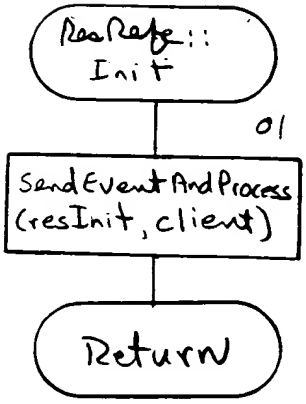



Fig 22

09252852.050699

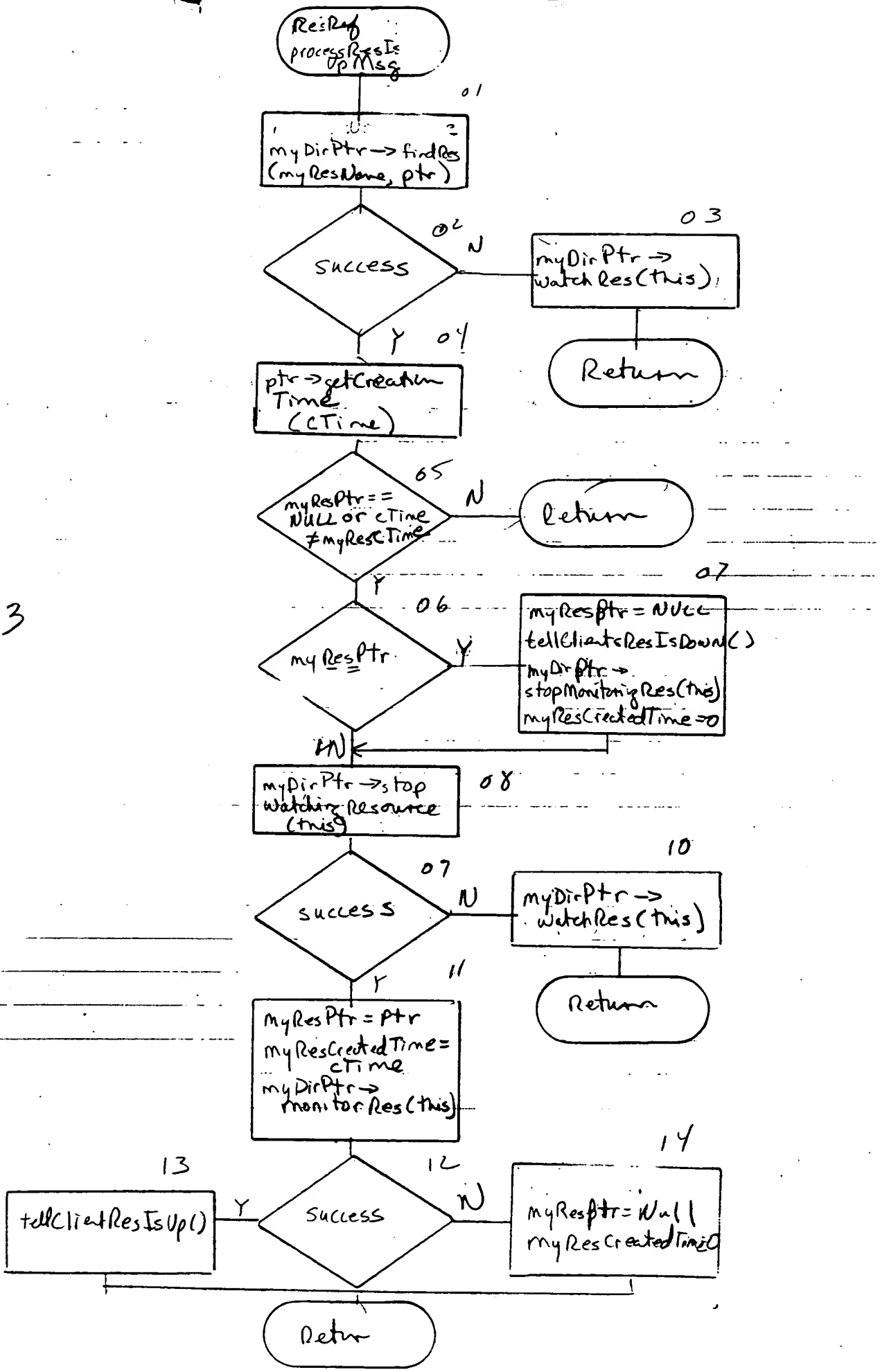
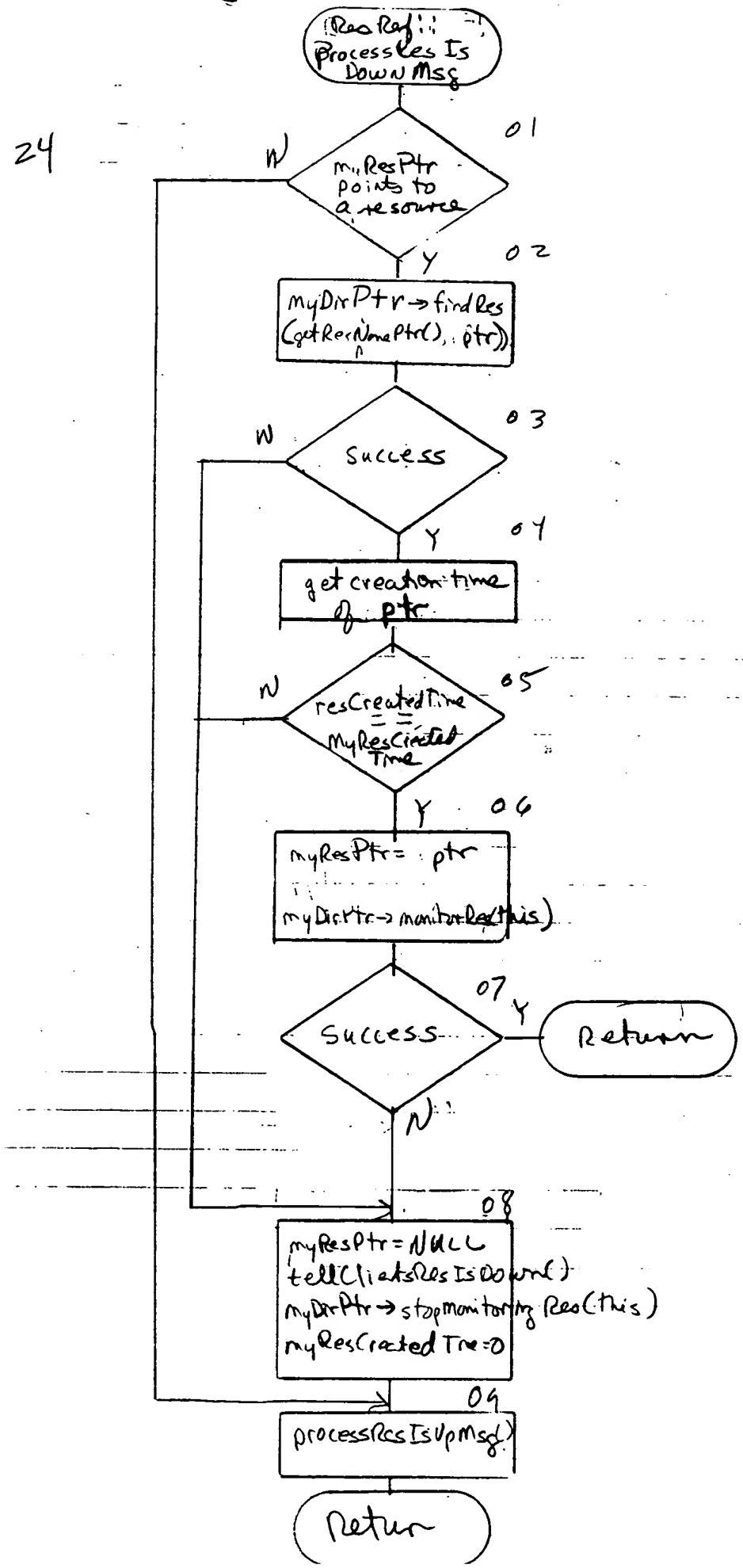



Fig 23

09220852.050999

Fig 24


ResRef::
ProcessRes Is
DownMsg
01
N
myResPtr
points to
a resource
Y
02
myDirPtr->findRes
(getResNamePtr(), ptr))
03
N
Success
Y
04
get creation time
of ptr
05
N
resCreatedTime
MyResCreated
Time
Y
06
myResPtr= ptr
myDirPtr->monitorRes(this)
07
Success
Y
Return
N
08
myResPtr=NULL
tellClientsResIsDown()
myDirPtr->stopMonitoringRes(this)
myResCreatedTime=0
09
processResIsUpMsg()
Return

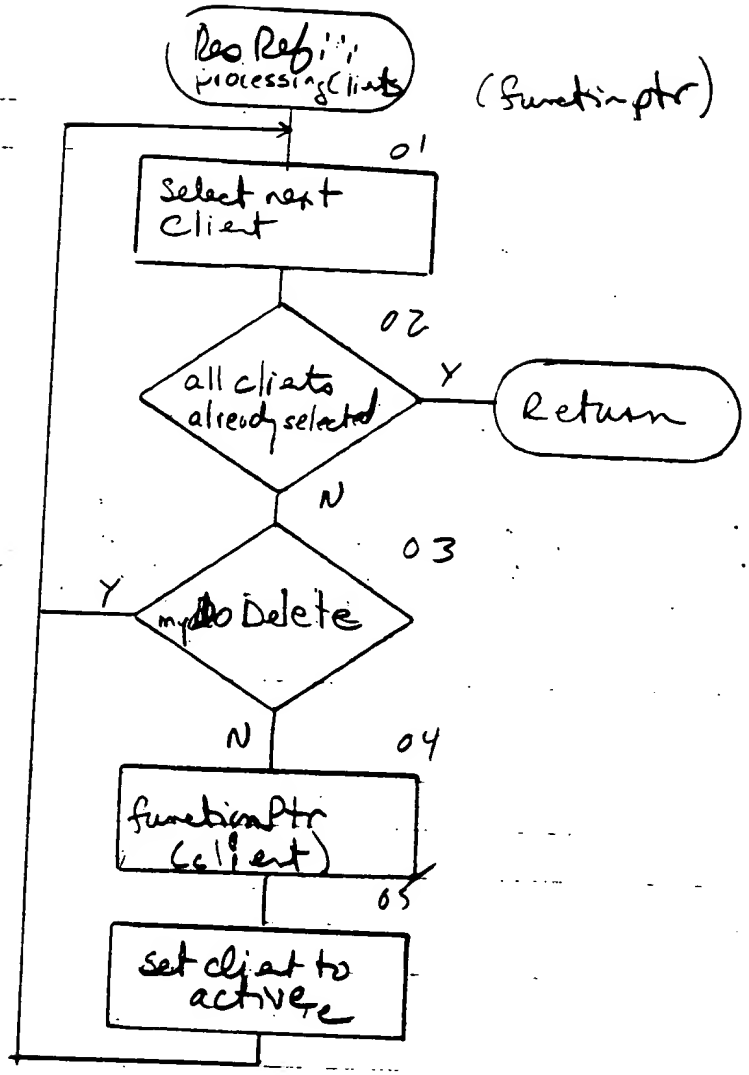

Rec Refining processing Clients
(function ptr)
01
Select next client
02
all clients already selected
Y
Return
N
03
Y
my Do Delete
N
04
function ptr (client)
05
set client to active

Fig 25

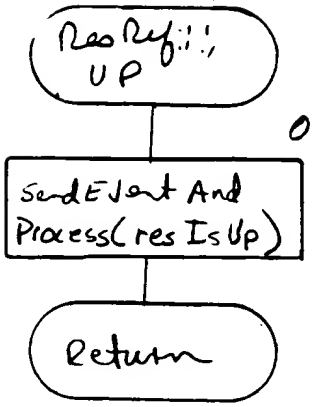



Fig 24

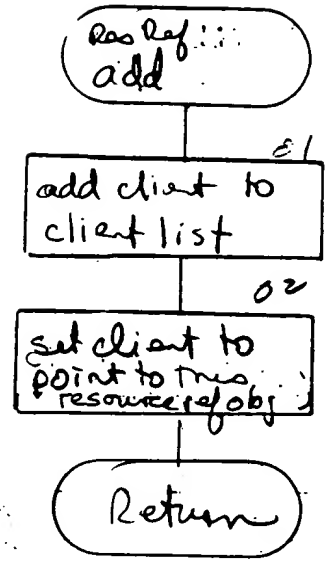

ResRef::
add
(client)
01
add client to
client list
02
set client to
point to this
resource ref obj
Return

Fig 27

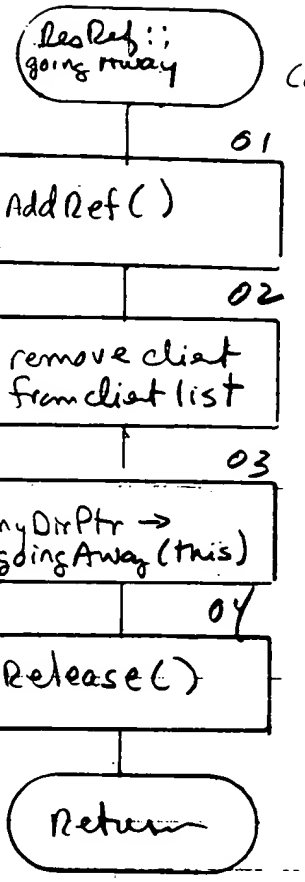

DesRef:: going away
(client)
01
AddRef()
02
remove client from client list
03
myDirPtr → goingAway(this)
04
Release()
Return

Fig 28

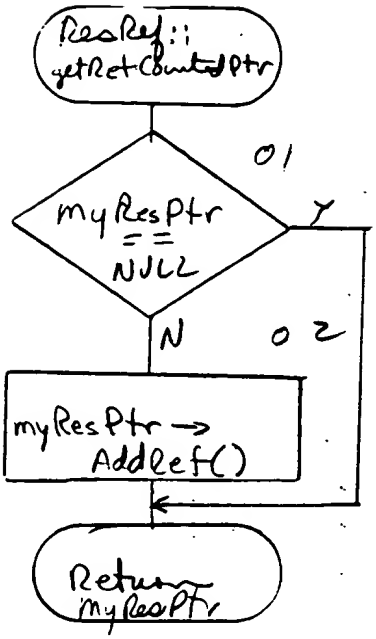

ResRef::
getRefCountPtr
01
myResPtr
==
NULL
Y
N
02
myResPtr →
AddRef()
Return
myResPtr

Fig 29.

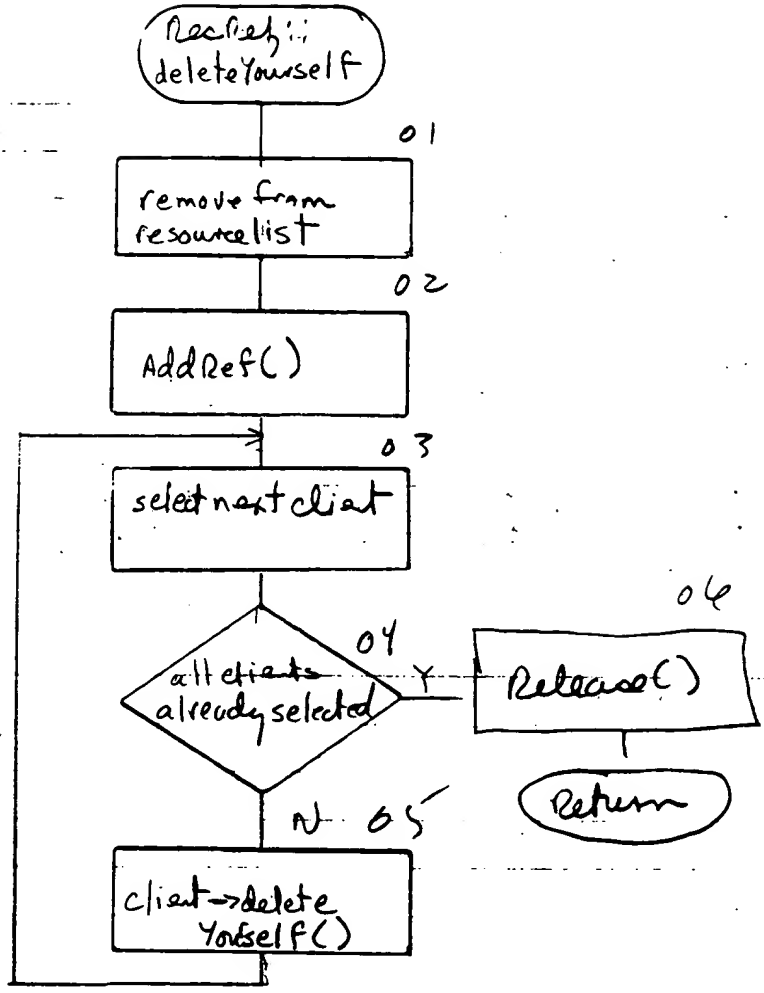

ReclaimObj::
deleteYourself
01
remove from
resourcelist
02
AddRef()
03
select next client
04
all clients
already selected
Y
06
Release()
Return
N
05
client->delete
YourSelf()

Fig 30

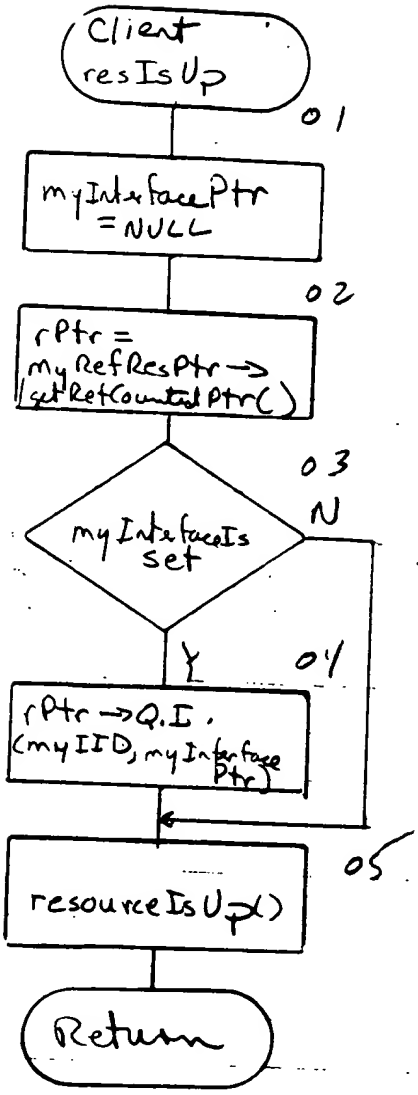

Client resIsUp
01
myInterfacePtr = NULL
02
rPtr = myRefResPtr -> getRefCountedPtr()
03
myInterfaceIs set
N
Y
04
rPtr -> Q.I. (myIID, myInterfacePtr)
05
resourceIsUp()
Return

Fig 31

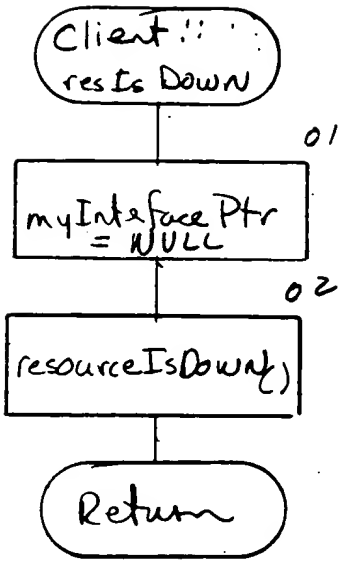

Client::
resIsDown
01
myInterfacePtr
= NULL
02
resourceIsDown()
Return

Fig 32

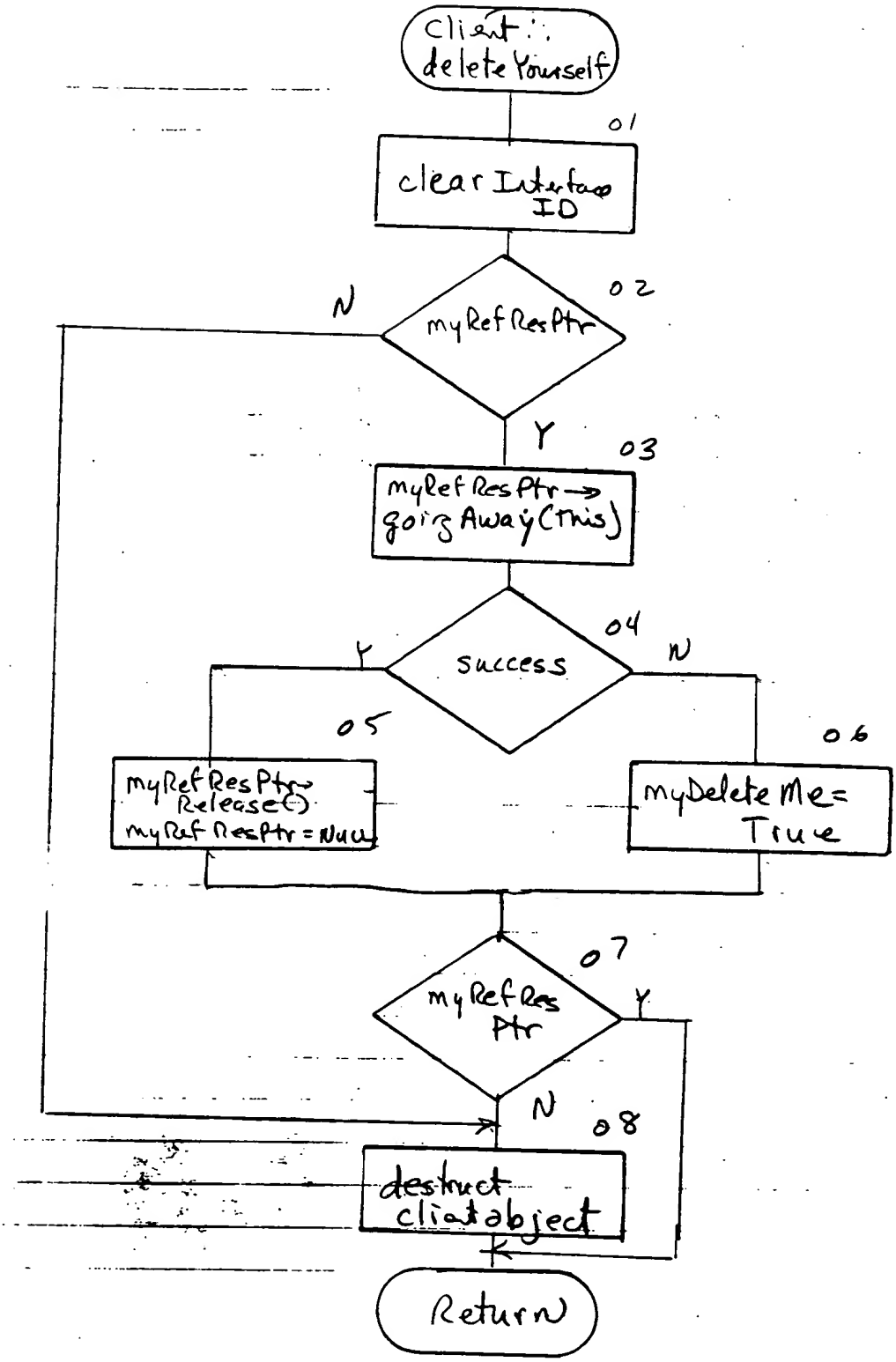
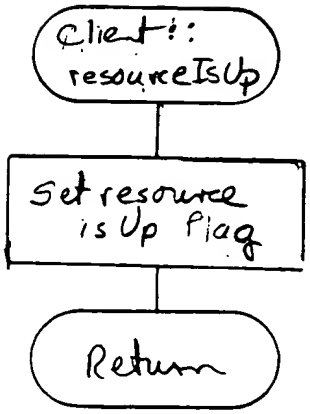

Client::
deleteYourself
01
clearInterface
ID
N
02
myRefResPtr
Y
03
myRefResPtr->
goingAway(This)
04
Y
success
N
05
myRefResPtr->
Release()
myRefResPtr=Null
06
myDeleteMe=
True
07
myRefRes
Ptr
Y
N
08
destruct
cliatobject
Return

Figure 33






6888250 0088250

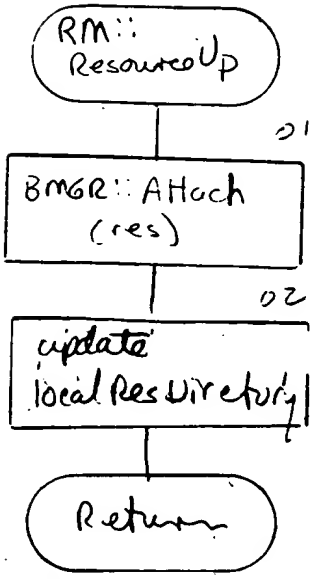

RM::
ResourceUp
(res)
01
BMGR::AHoch
(res)
02
update
local ResDirectory
Return

Fig 35

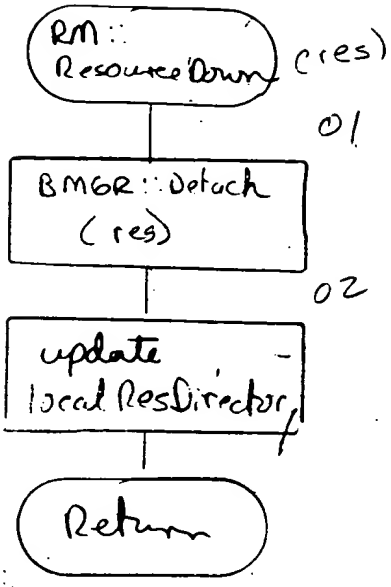

RM::
ResourceDown
(res)
01
BMGR::Detach
(res)
02
update
local ResDirector
Return

Fig 36

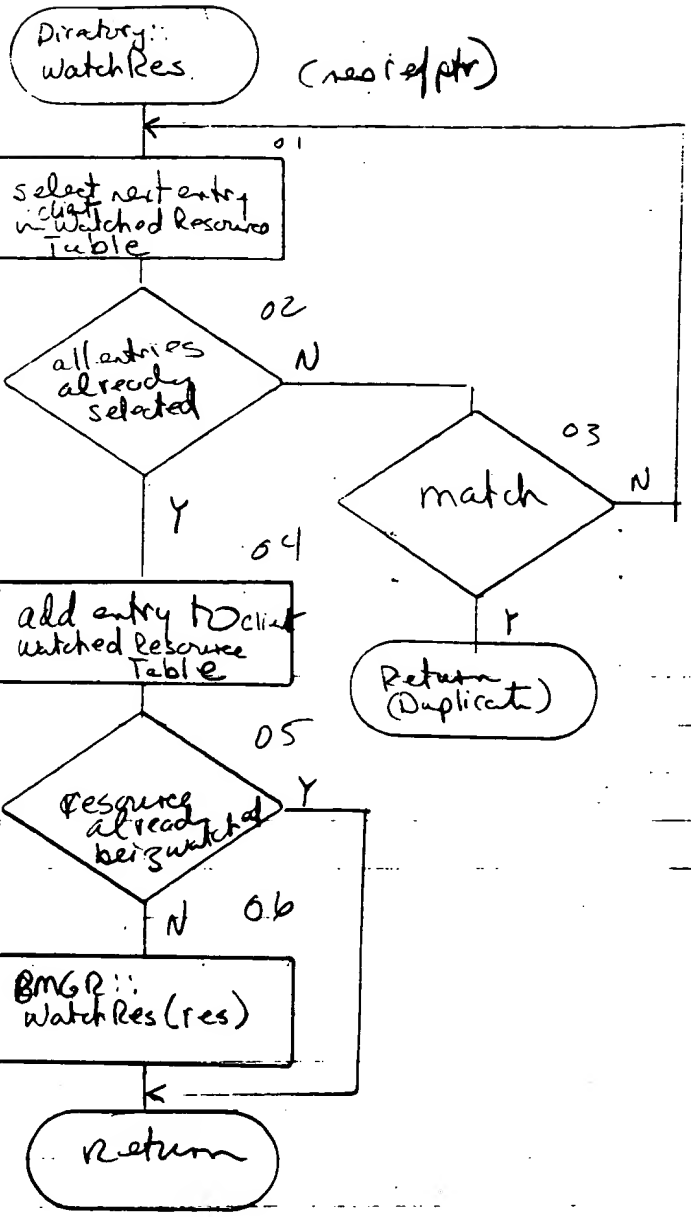

Directory::
WatchRes
(resid/ptr)
01
select next entry in client Watched Resource Table
02
all entries already selected
N
Y
03
match
N
Y
04
add entry to client watched Resource Table
Return (Duplicate)
05
Resource already being watched
Y
N
06
BMGR::
WatchRes(res)
Return

Fig 37

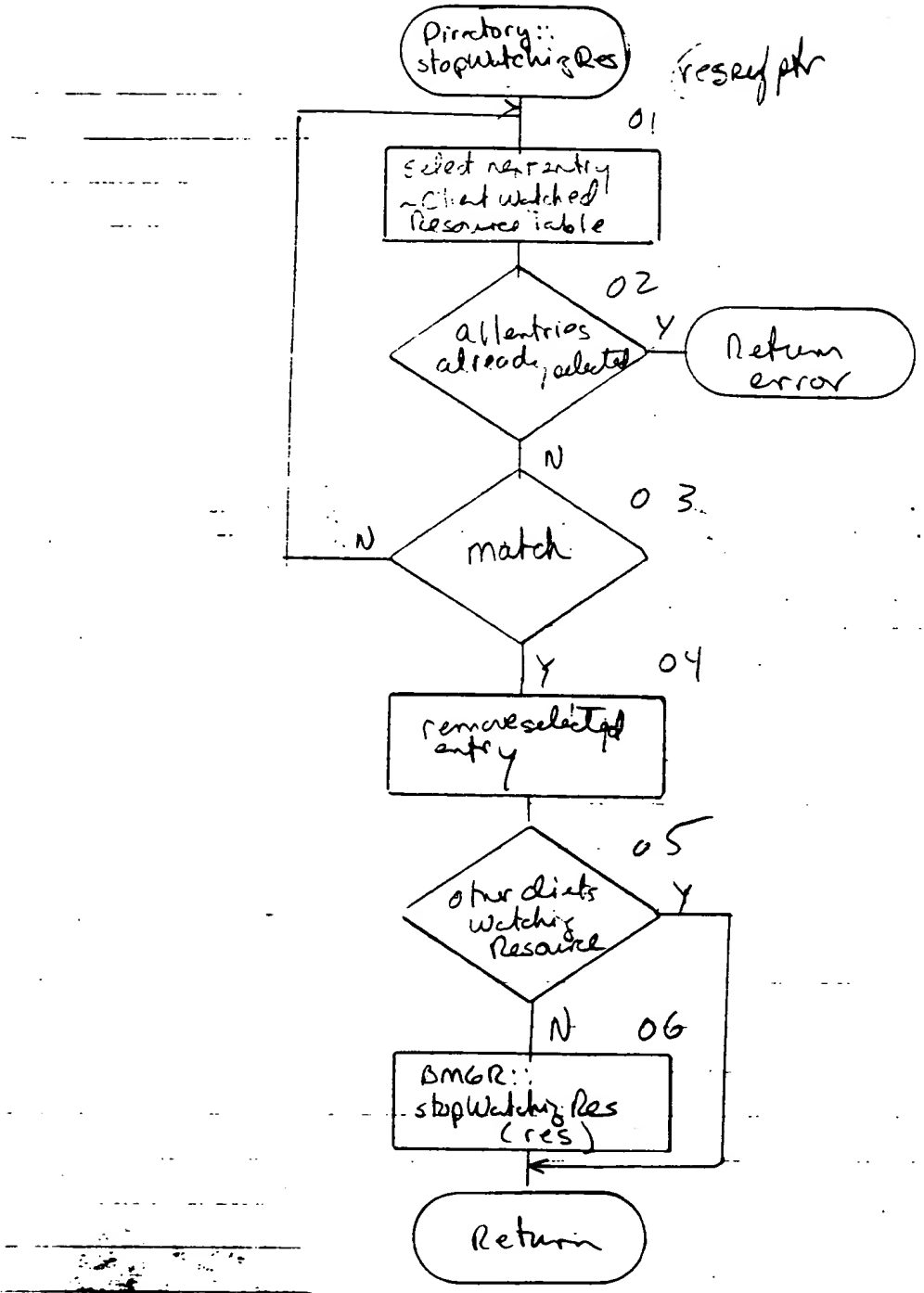

Directory::
stopWatchingRes
resRef ptr
01
select next entry
~Client watched
Resource table
02
all entries
already selected
Y
Return
error
N
03
match
N
Y
04
remove selected
entry
05
other clients
watching
Resource
Y
N
06
BMGR::
stopWatchingRes
(res)
Return

Fig 38

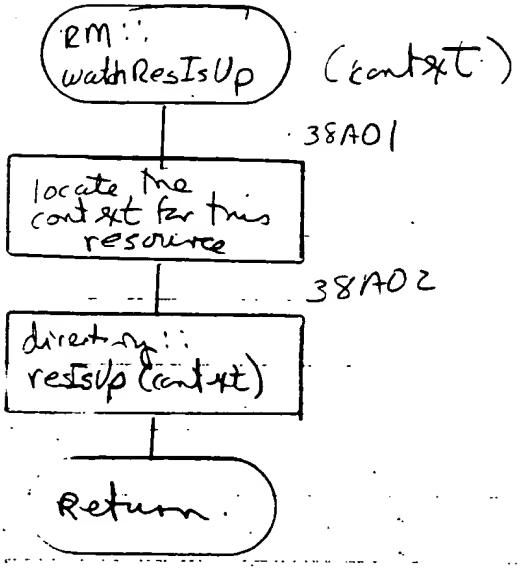

RM::
watchResIsUp
(context)
38A01
locate the
context for this
resource
38A02
directory::
resIsUp (context)
Return.

Figure 38A

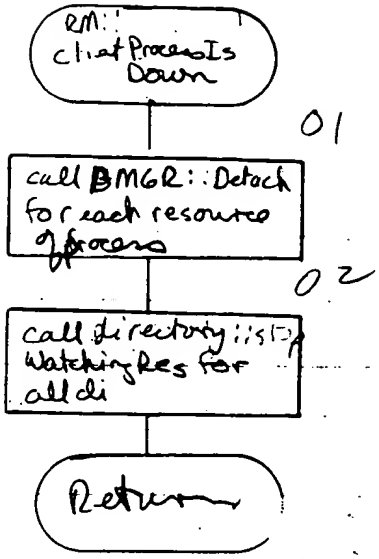
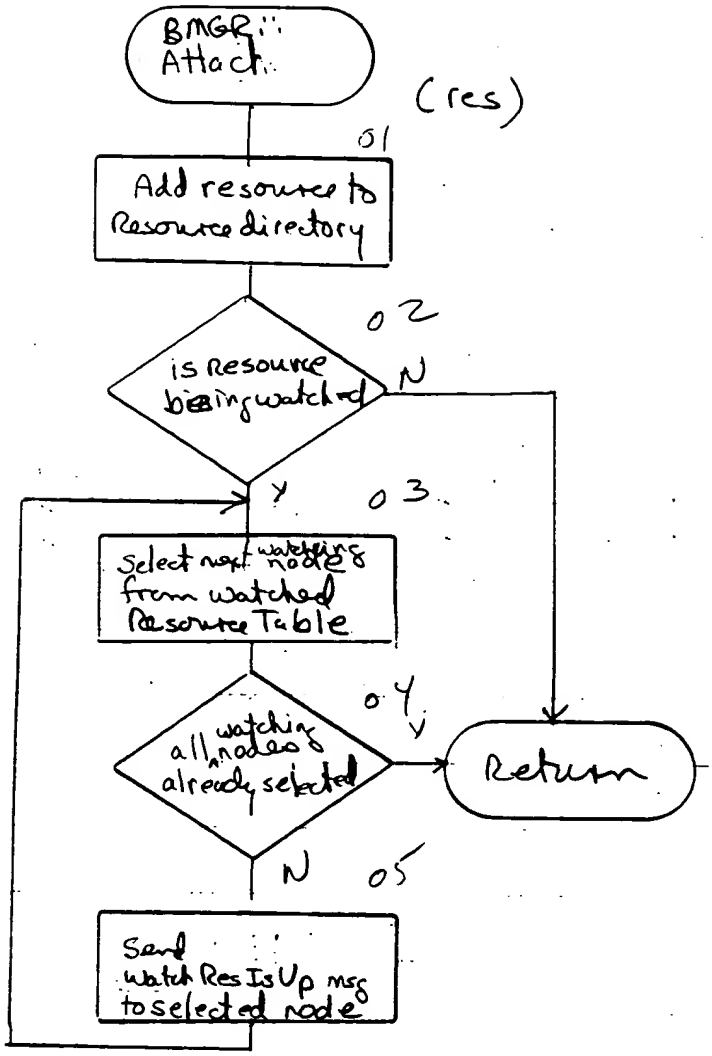

RM::
clientProcessIs
Down
01
call BMGR::Detach
for each resource
of process
02
call directory::isIP
WatchingRes for
all di
Return

Fig 39


BMGR:: Attach (res)
01
Add resource to Resource directory
02
is Resource being watched
N
Y
03
Select next watching node from watched Resource Table
04
all watching nodes already selected
Y
Return
N
05
Send watch Res Is Up msg to selected node

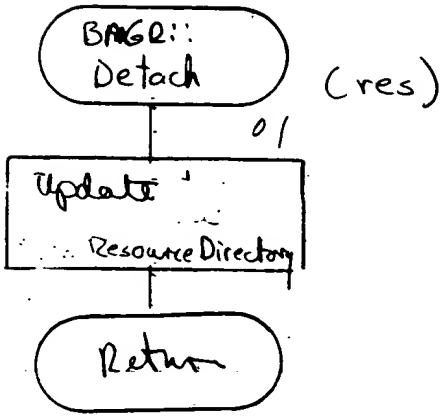



Fig 41

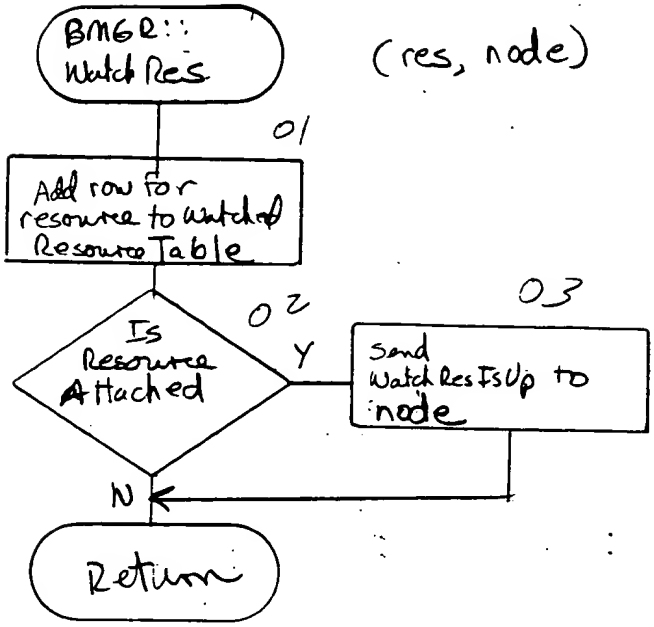

BMGR::
Watch Res
(res, node)
01
Add row for resource to watched Resource Table
02
Is Resource Attached
Y
03
Send Watch Res Is Up to node
N
Return

Fig 42

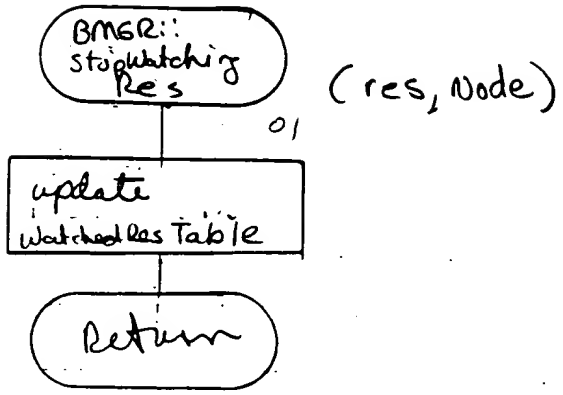
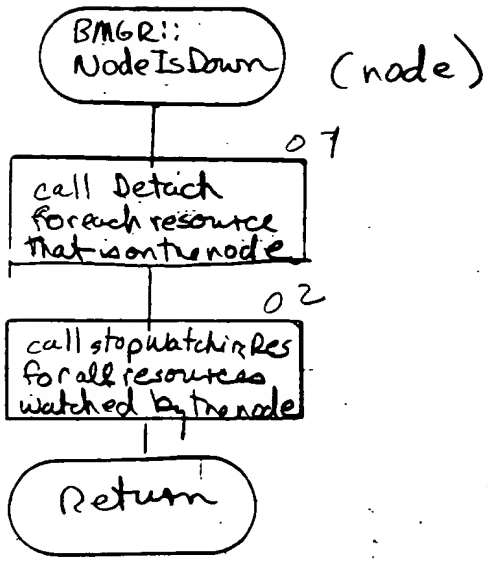

BMGR::
stopWatching
Res
(res, Node)
01
update
WatchedRes Table
Return

Fig 13


BMGR::
NodeIsDown
(node)
01
call Detach
foreach resource
that is on the node
02
call stopWatching Res
for all resources
watched by the node
Return

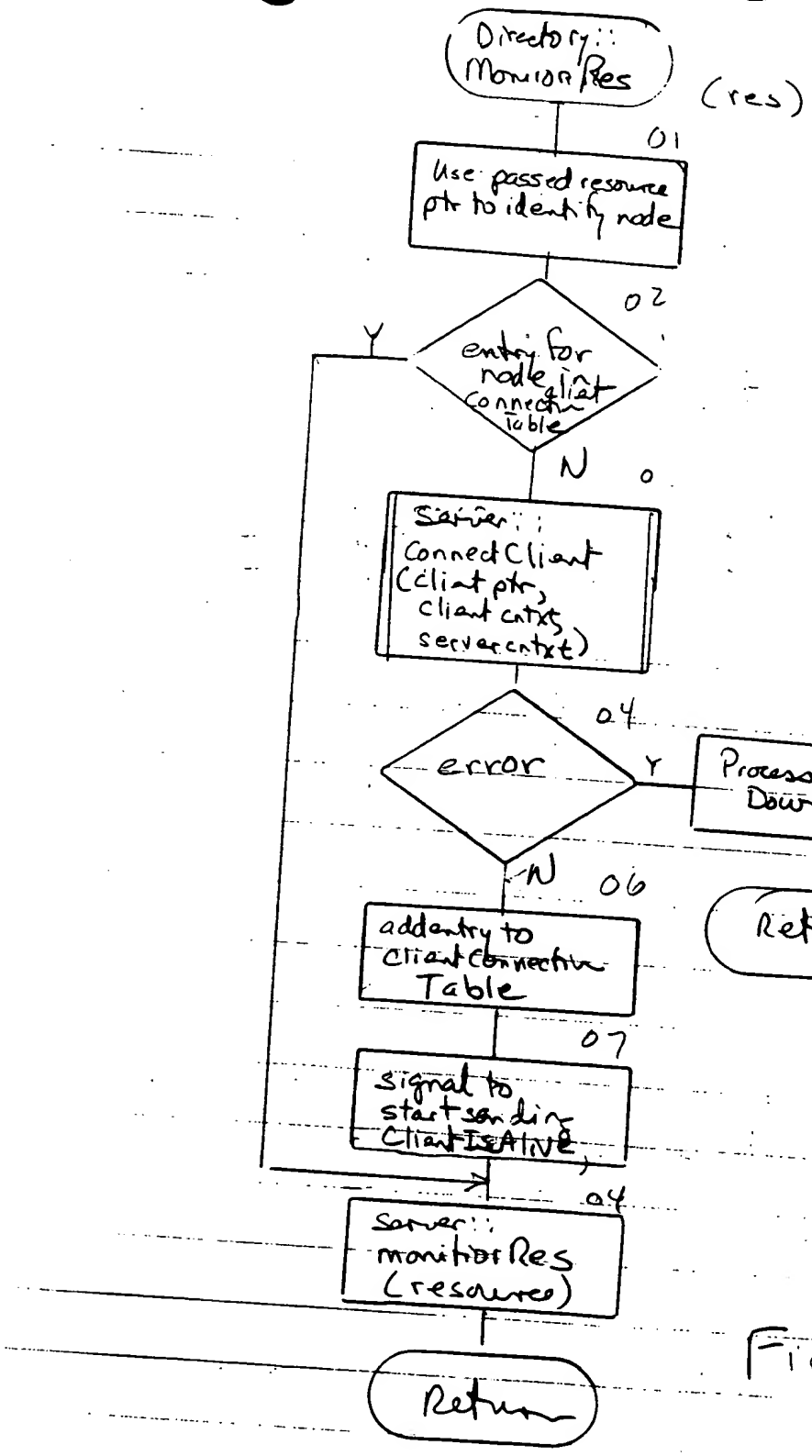

Directory::
MonitorRes
(res)
01
Use passed resource
ptr to identify node
02
entry for
node & client
connection
table
Y
N
0
Server::
ConnectClient
(Cclient ptr,
client cntxt,
servercntxt)
04
error
Y
05
Process Server Is
Down
Return
N
06
addentry to
client Connection
Table
07
signal to
start sending
ClientIsALIVE
04
Server::
monitorRes
(resource)
Return

Fig 45

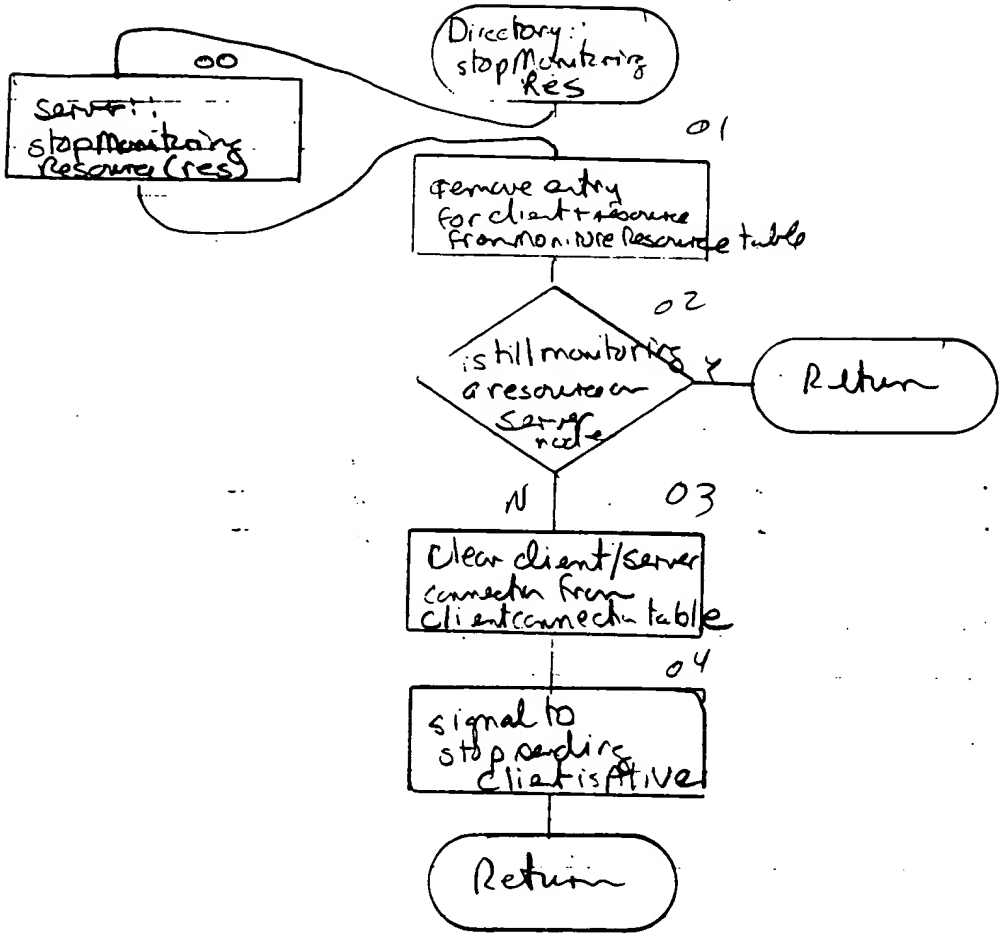

Directory: stopMonitoring Res
00
Server: stopMonitoring Resource(res)
01
Remove entry for client+resource from monitored resource table
02
is still monitoring a resource on server node
Y
Return
N
03
Clear client/server connection from clientconnection table
04
signal to stop sending client is Active
Return

Fig 46

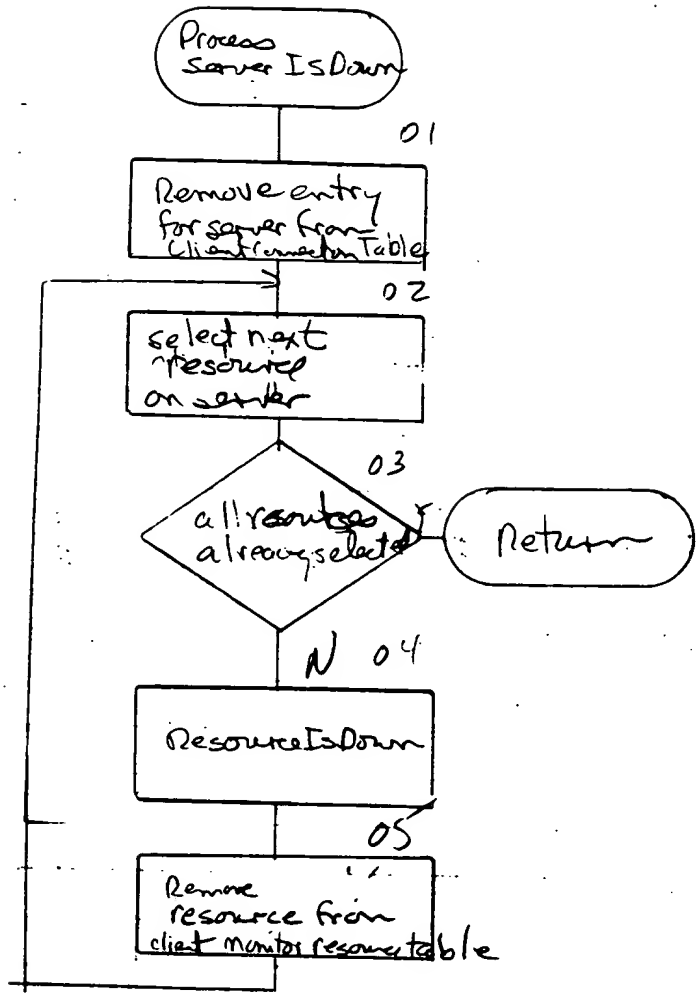

Process Server Is Down
01
Remove entry for server from Client Connection Table
02
select next resource on server
03
all resources already selected
Y
Return
N
04
Resource Is Down
05
Remove resource from client monitor resource table

Fig 46A

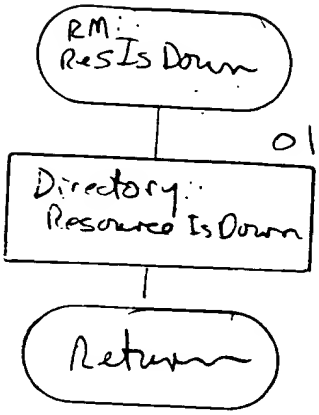
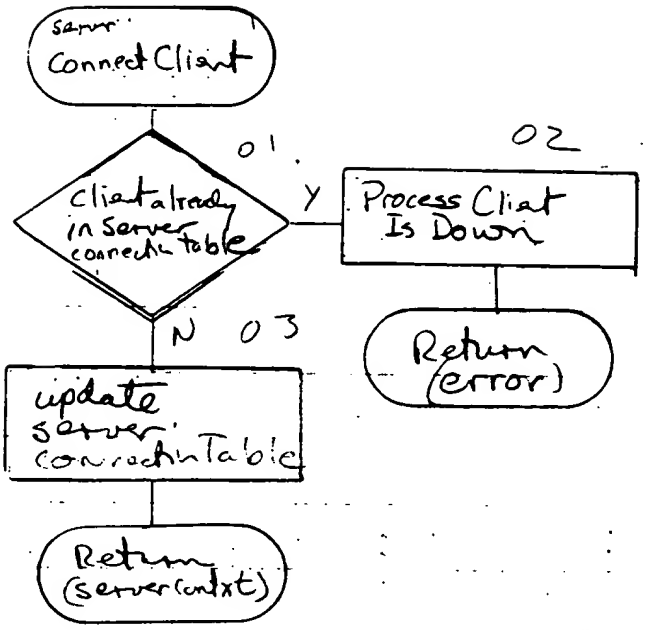



Fig 47

5,000,000






0520352.0520360

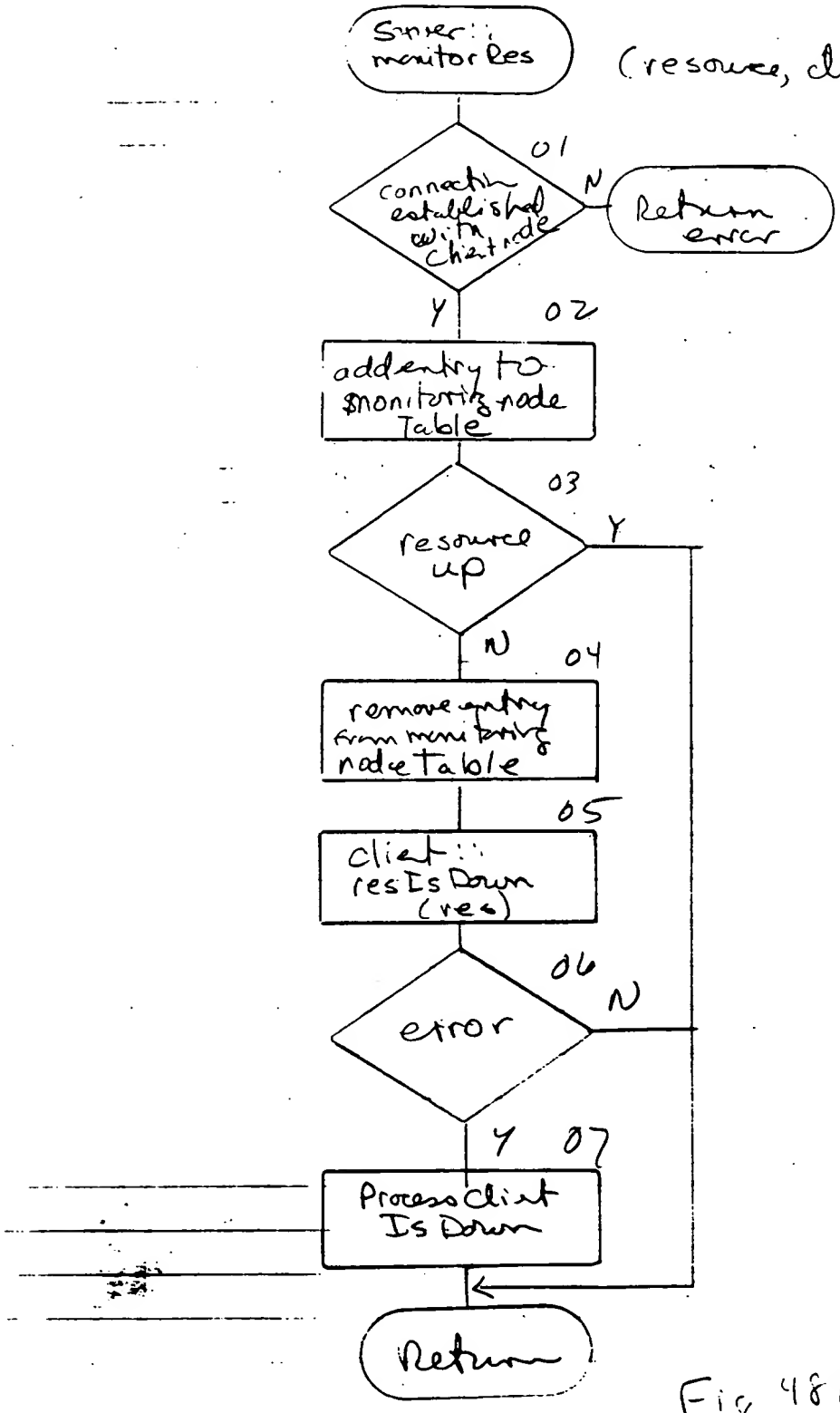

Server::
monitorRes
(resource, client, context)
01
connection established with client node
N
Return error
Y
02
add entry to monitoring node Table
03
resource up
Y
N
04
remove entry from monitoring node Table
05
Client::
resIsDown
(res)
06
error
N
Y
07
Process Client Is Down
Return

Fig 48A

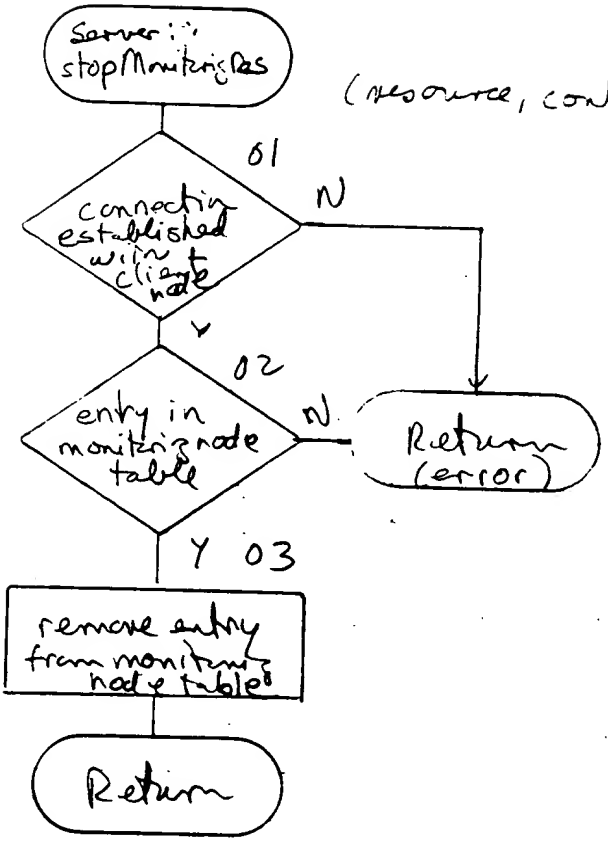
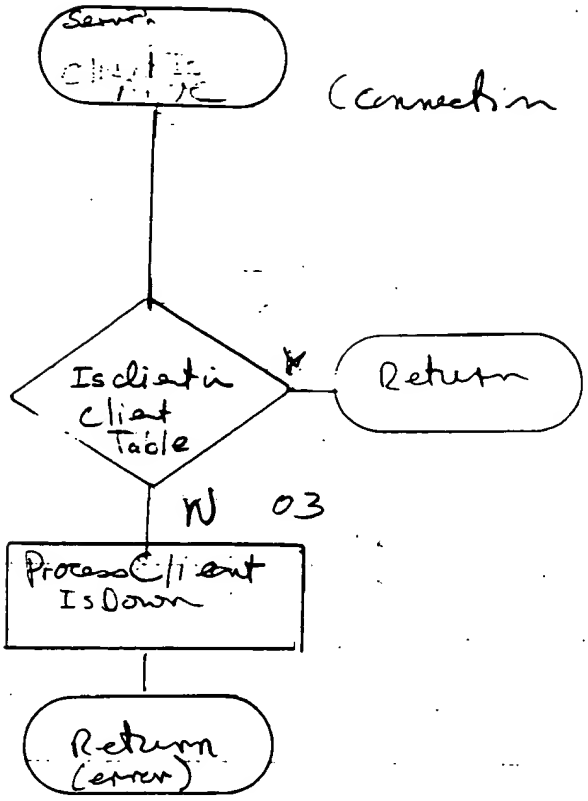
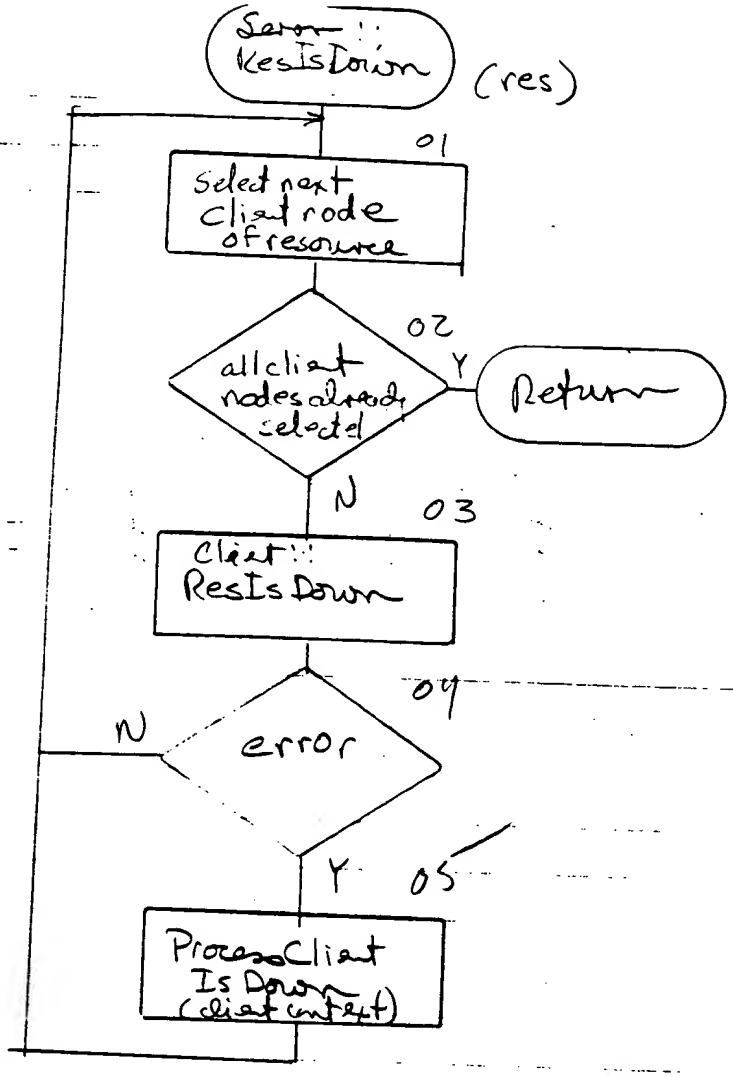
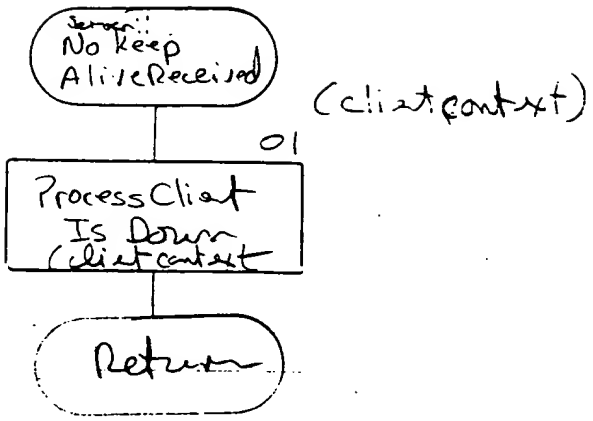

Server::
stopMonitoringRes
(resource, context)
01
connection established with client node
N
Y
02
entry in monitoring node table
N
Return (error)
Y 03
remove entry from monitoring node table
Return

Figure 48B


Server
Client
Connection
Is client in client Table
N
Return
03
Process Client Is Down
Return (error)

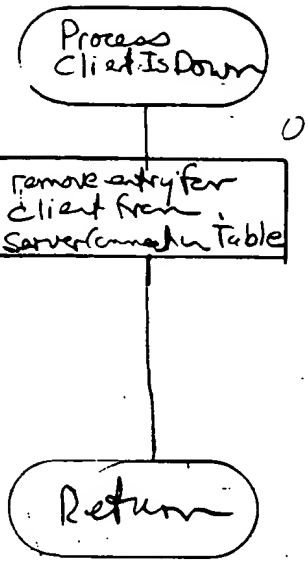

Server::
ResIsDown (res)
01
Select next
Client node
of resource
02
all client
nodes already
selected
Y
Return
N
03
Client::
ResIsDown
04
error
N
Y
05
Process Client
Is Down
(client context)

50

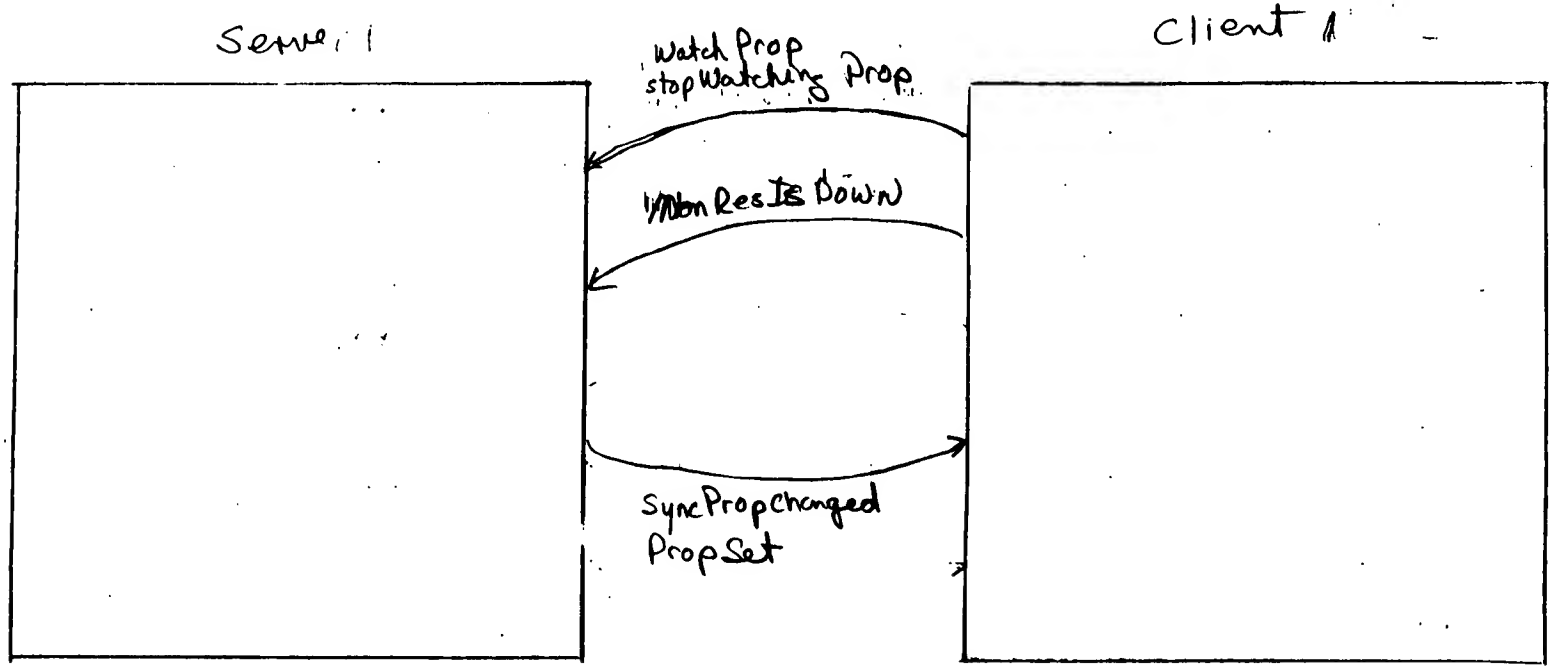





0050050-0050051

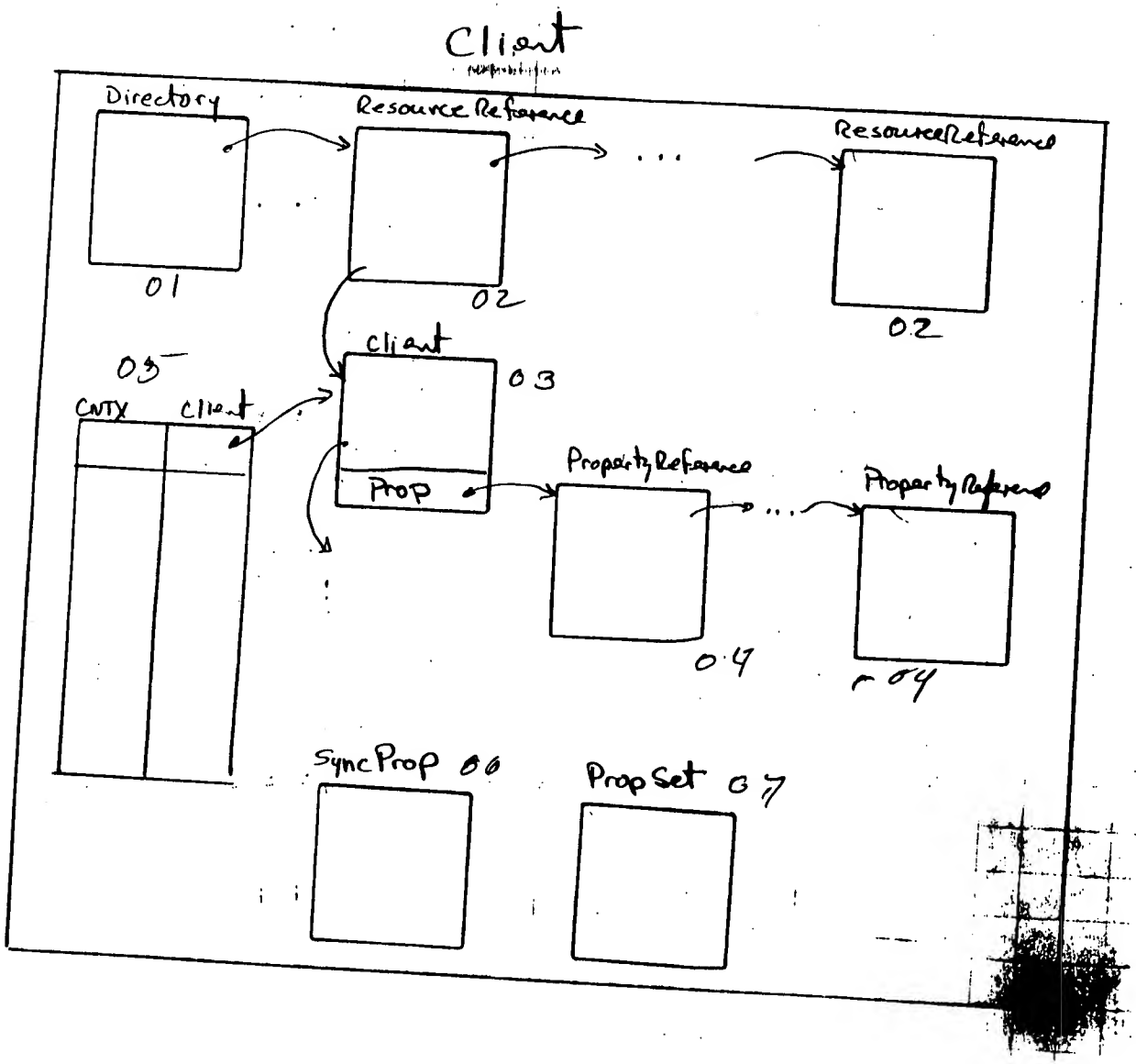

Process ClientIsDown
(clientcontext)
02
Remove entry for client from ServerConnectionTable
Return

52



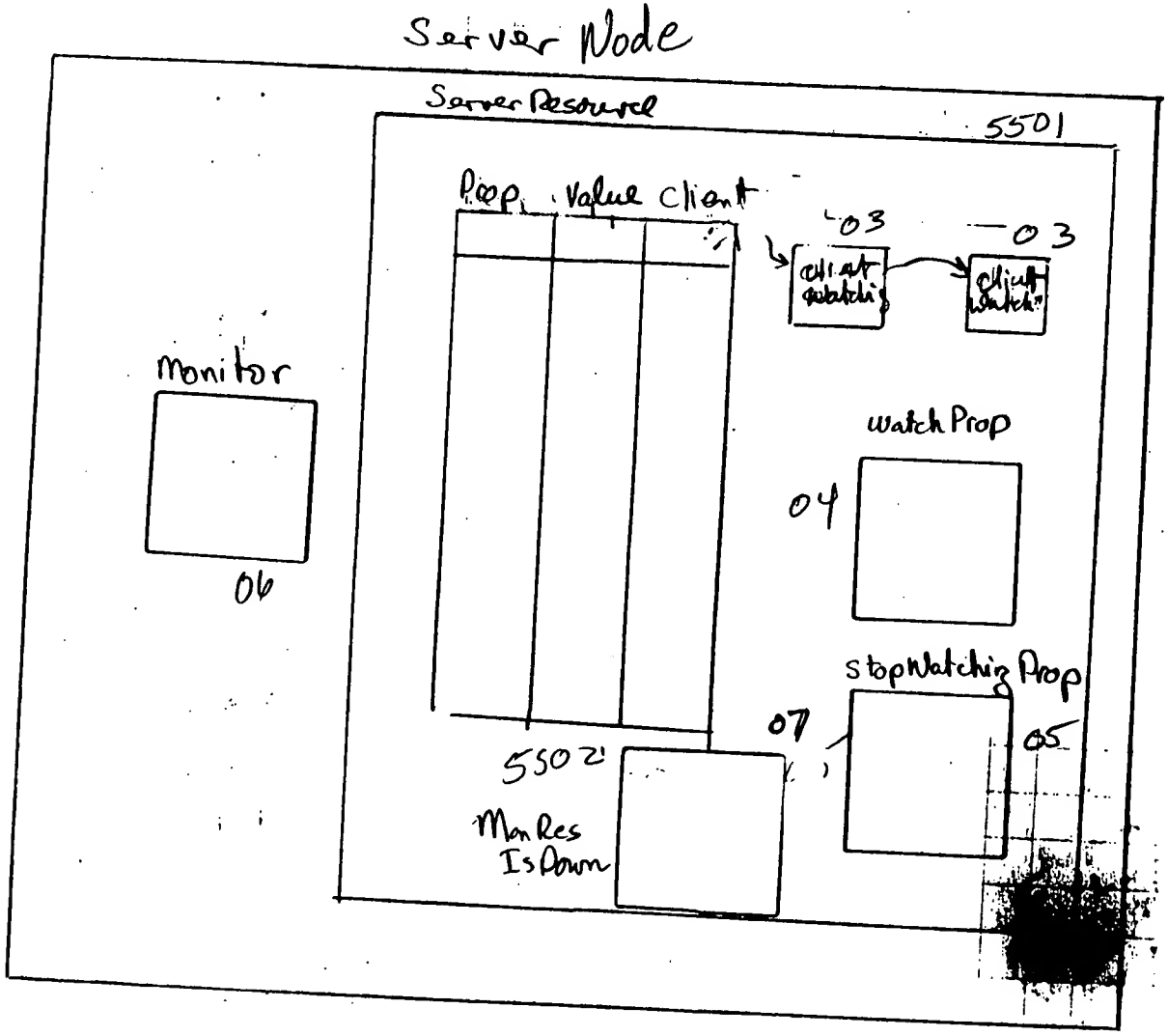



0932852.052899



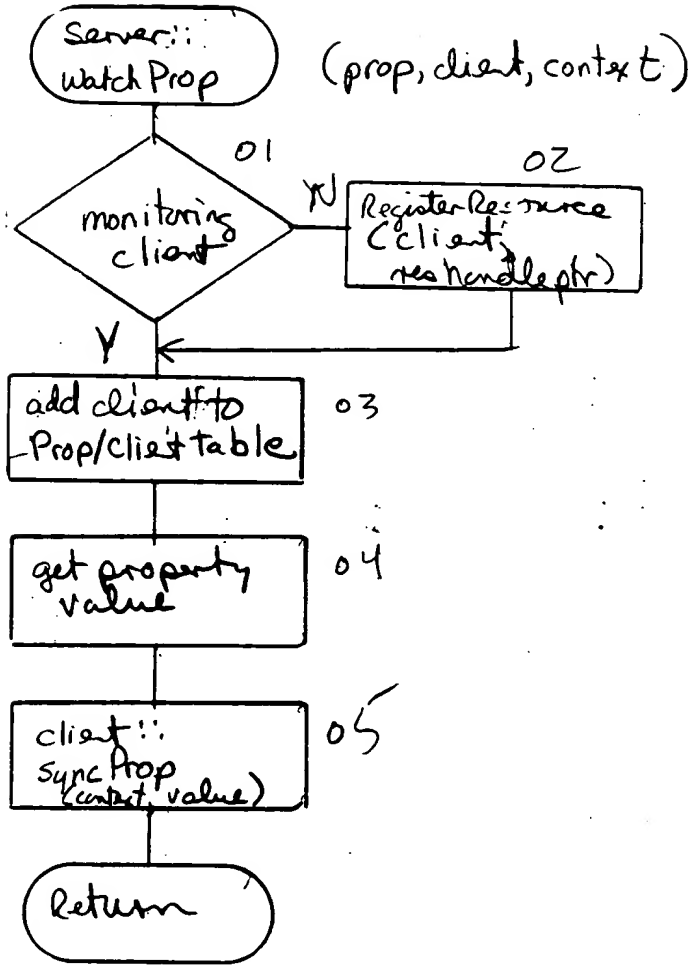


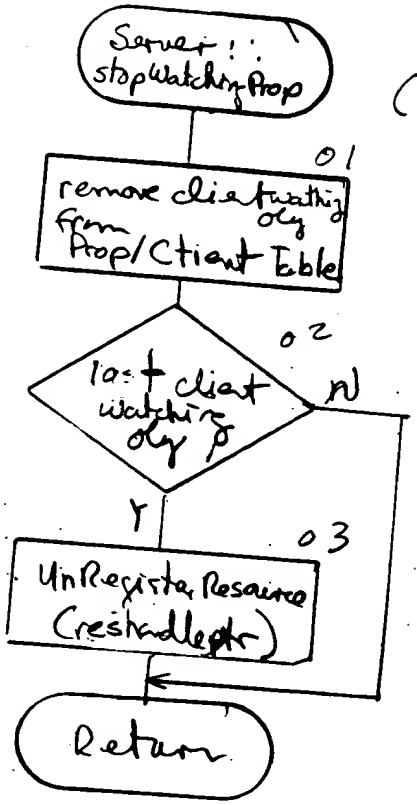

09328852.052899





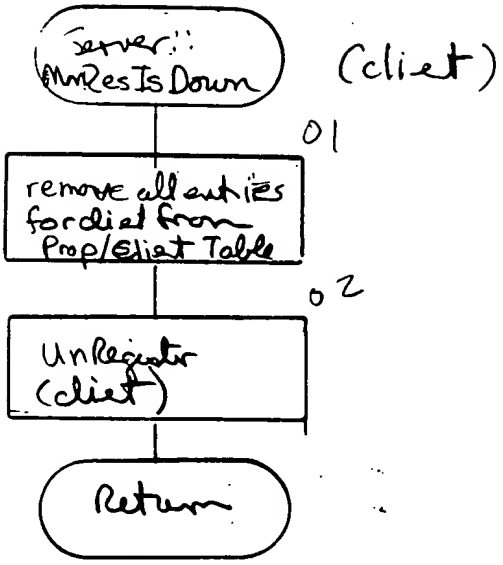

09322852.052599


Server::
WatchProp
(prop, client, context)
01
monitoring client
N
02
RegisterResource (client, resHandlePtr)
Y
add client to Prop/Client table
03
get property value
04
client::
SyncProp (context, value)
05
Return



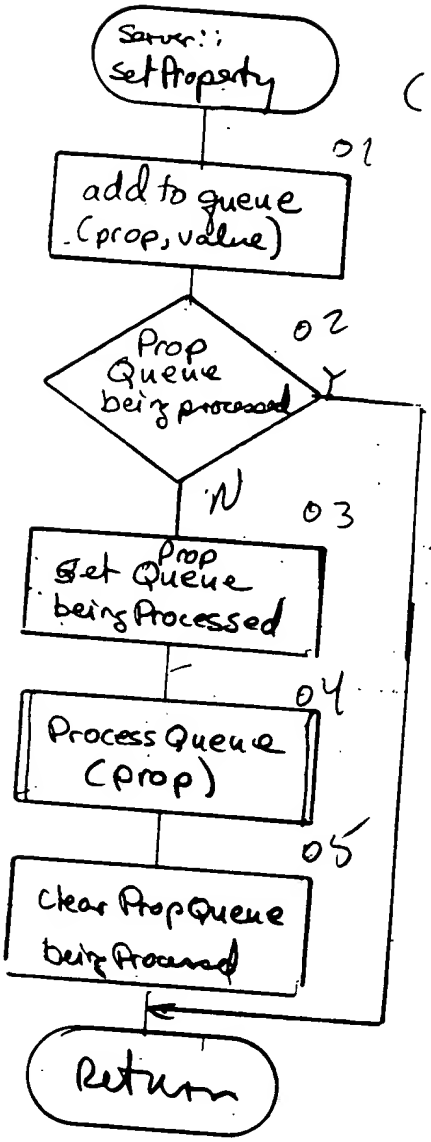


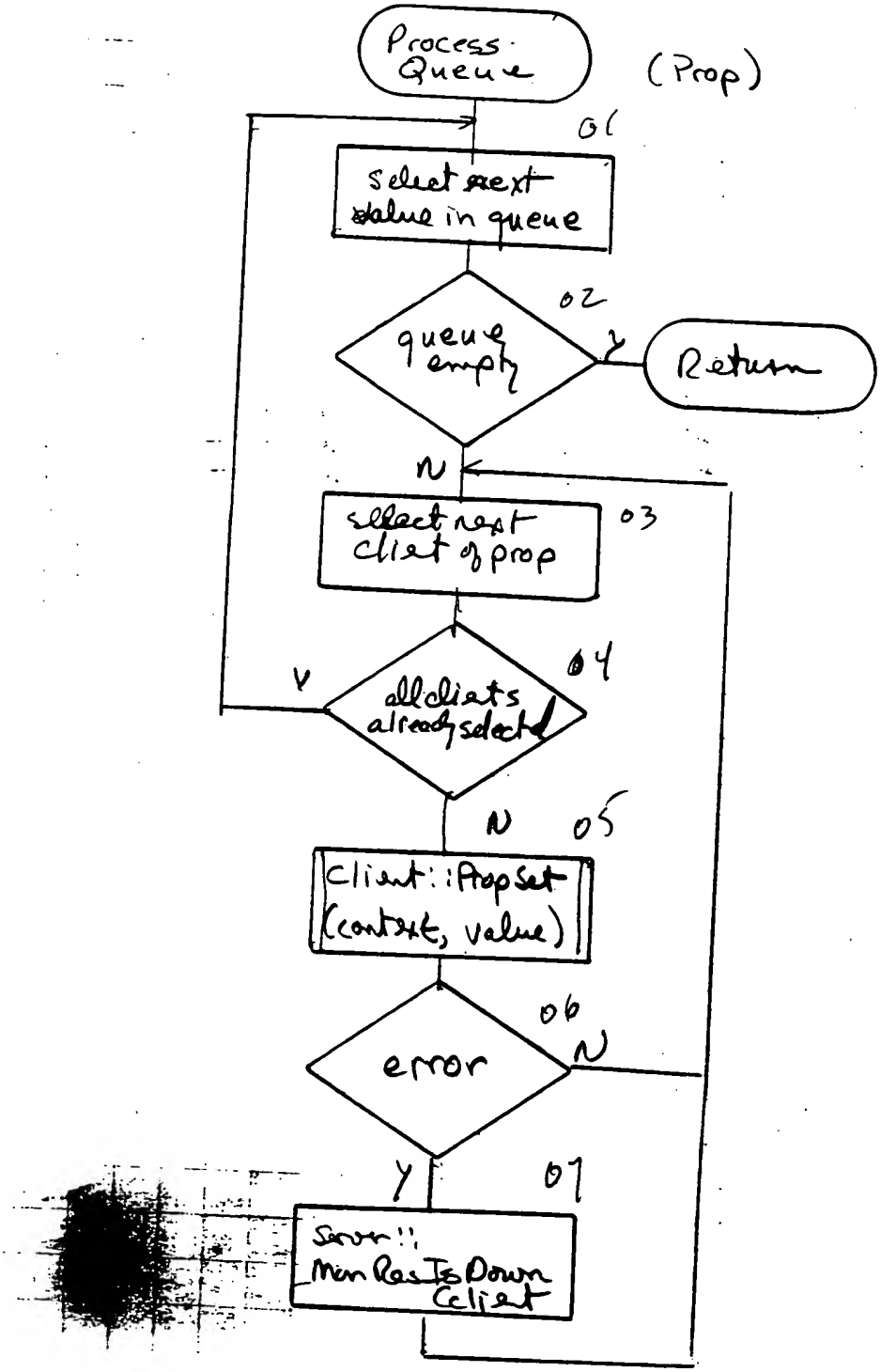

0050050.050050



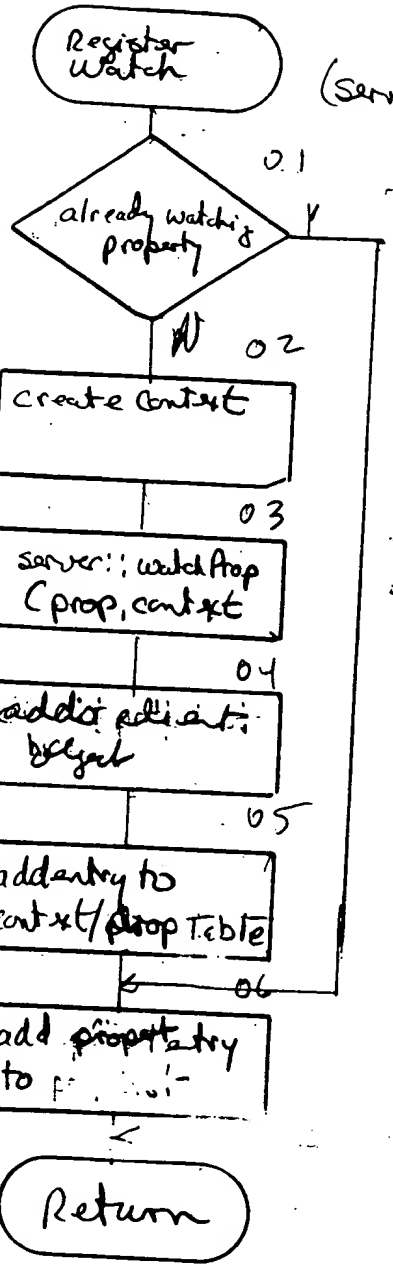


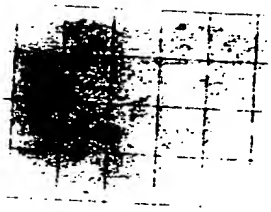

0050050.0050050

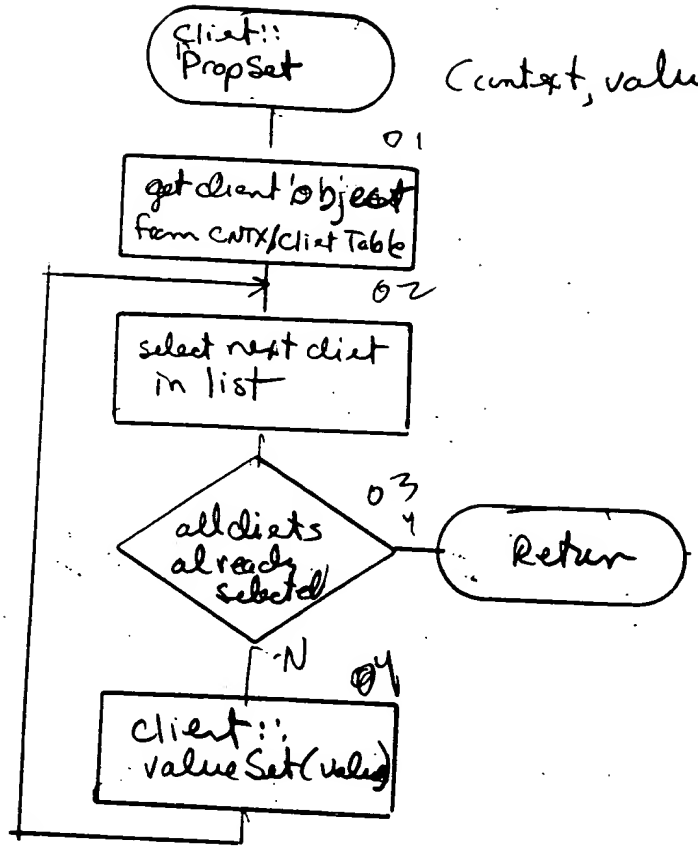

Server::
SetProperty
(prop, value)
01
add to queue
(prop, value)
02
Prop
Queue
being processed
Y
N
03
Set Prop Queue
being Processed
04
Process Queue
(prop)
05
Clear PropQueue
being Processed
Return


Process Queue
(Prop)
01
select next value in queue
02
queue empty
Y
Return
N
select next client of prop
03
all clients already selected
04
Y
N
05
Client::PropSet (context, value)
error
06
N
Y
07
Server:: Mem Res Is Down Client

60


Register Watch
(server, prop, client)
01
already watching property
Y
N
02
create context
03
server::watchProp (prop, context
04
addClient to object
05
add entry to context/prop Table
06
add property entry to
Return








0522052.0522053

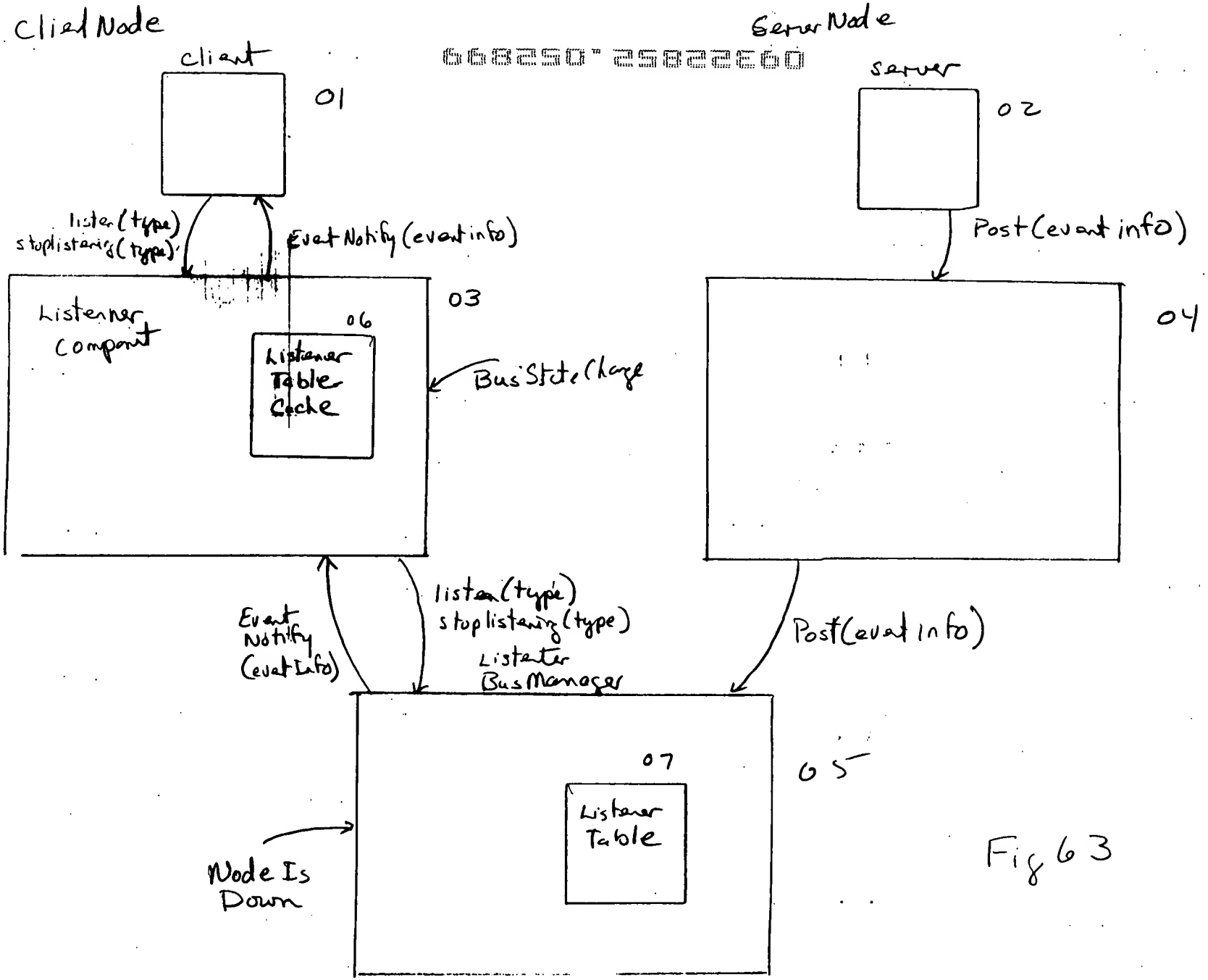

Client Node
Server Node
client
01
server
02
listen(type)
stoplistening(type)
Event Notify (event info)
Post(event info)
Listener Component
03
06
Listener Table Cache
Bus State Change
04
listen(type)
stoplistening(type)
Event Notify (event Info)
Post(event info)
Listener Bus Manager
07
05
Listener Table
Node Is Down

Fig 6 3